TUGAS AKHIR – TE 141599

PERANCANGAN KONTROLER PID-GA UNTUK SISTEM PENGATURAN *LEVEL* DAN *PRESSURE* PADA SIMULATOR *PLANT BOILER* DENGAN METODE DEKOPLING SISTEM MIMO

Aryani Fabiany
NRP 2212100191

Dosen Pembimbing
Eka Iskandar, ST., MT.
Ir. Rusdhianto Effendi A.K., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – TE 141599

DESIGNING PID-GA CONTROLLER TO CONTROL LEVEL AND PRESSURE OF PLANT SIMULATOR BOILER WITH DECOUPLING SYSTEM MIMO

Aryani Fabiany
NRP 2212100191

Supervisor
Eka Iskandar, ST., MT.
Ir. Rusdhianto Effendi A.K., MT.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

PERANCANGAN KONTROLER PID-GA UNTUK SISTEM PENGATURAN *LEVEL* DAN *PRESSURE* PADA SIMULATOR *PLANT BOILER* DENGAN METODE DEKOPLING SISTEM MIMO

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Pengaturan
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Eka Iskandar, S.T., M.T.	**Ir. Rusdhianto Effendi A.K., M.T.**
NIP. 1980 05 28 2008 12 1001	**NIP. 1957 04 24 1985 02 1001**

SURABAYA
JUNI, 2016

PERANCANGAN KONTROLER PID-GA UNTUK SISTEM PENGATURAN *LEVEL* DAN *PRESSURE* PADA SIMULATOR *PLANT BOILER* DENGAN METODE DEKOPLING SISTEM MIMO

Aryani Fabiany – 2212100191

Pembimbing : 1. Eka Iskandar, ST., MT.
2. Ir. Rusdhianto Effendie A.K.,MT.

ABSTRAK

Sistem pengendalian di industri merupakan faktor yang sangat penting dalam proses produksi. Boiler merupakan salah satu bagian pada industri yang memiliki peran vital.

Sistem nonlinear yang terdapat pada pemodelan Boiler mempengaruhi kestabilan sistem. Selain itu konfigurasi sistem Multi Input Multi Output membuat variabel yang diatur mengalami kesulitan untuk mengikuti sinyal referensi.

Dekopling digunakan untuk mengatur konfigurasi MIMO dan kontrol PID untuk mengatur level dan pressure pada Boiler. Algoritma genetika digunakan untuk tuning parameter PID yang paling optimal. Berdasarkan hasil simulasi didapatkan parameter PID Kp = 0,1451, Ki =9,9170e-05, Kd = 4,7844e-14 untuk pressure dengan overshoot 5,8% dan Kp = 0,2624, Ki =7,6734e-09, Kd = 1,3795e-17 untuk level.

Kata kunci : Boiler, dekopling, Kontroler PID, Algoritma Genetika

DESIGNING PID-GA CONTROLLER TO CONTROL LEVEL AND PRESSURE OF PLANT SIMULATOR BOILER WITH DECOUPLING SYSTEM MIMO

Aryani Fabiany – 2212100191

Supervisor : 1. Eka Iskandar, ST., MT.
2. Ir. Rusdhianto Effendie A.K.,MT.

ABSTRACT

The control systems in the industry is a very important factor in the production process. Boiler is one part of the industry has a vital role.

Nonlinear systems modeling contained in Boiler affect the stability of the system. In addition the system configuration Multi Input Multi Output create a variable that is set a challenge to keep the reference signal.

Decoupling is used to set the MIMO configuration and control PID controller is used to set the level and pressure in the boiler. Genetic algorithm is used to tune PID parameters.. Based on simulation, the resulted PID parameters are Kp = 0,1451, Ki =9,9170e-05, Kd = 4,*7844e-14 for pressure with overshoot 5,8% dan Kp = 0,2624, Ki =7,6734e-09 , Kd = 1,3795e-17 for level.*

Keywords : *Boiler, Decoupling MIMO, PID Controller, Genetic Algorithm.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Sistem pengendalian di industri merupakan faktor yang sangat penting dalam proses produksi. Agar produksi tetap tercapai, maka suatu sistem pengendalian sangat diperlukan untuk menjaga kestabilan variabel proses. Variabel proses antara lain *temperature, pressure, flow, level,* konsentrasi, *volume* dan lain sebagainya.

Boiler-turbine merupakan salah satu bagian di industri yang memiliki fungsi yang sangat vital. Secara umum *Boiler* terdiri dari beberapa sistem, diantaranya adalah sistem *feedwater,* sistem *steam,* dan sistem *fuel* yang terintegrasi menjadi satu kesatuan. Sistem *feedwate*r berfungsi sebagai penyedia air untuk *Boiler-turbine* yang bekerja secara otomatis sesuai kebutuhan *steam* dan kemampuan dari *Boiler* itu sendiri. Sedangkan sistem *steam* berfungsi sebagai penyedia uap air untuk proses pada *plant* yang lain. Kedua sistem ini memerlukan suatu pengaturan agar kondisi air dan uap air dapat untuk memenuhi kebutuhan proses selanjutnya. Pengaturan air dilakukan dengan mengendalikan *level* air yang terdapat di dalam *Boiler-turbine* itu sendiri, sedangkan pengaturan uap air dilakukan dengan mengendalikan *pressure* dari *steam* yang merupakan hasil dari *Boiler* sendiri. *Pressure* yang dihasilkan oleh *Boiler* dipengaruhi oleh sistem *fuel* dan kondisi *level* air yang terdapat di dalam *Boiler-turbine.*

Pemodelan *Boiler-turbine* termasuk kategori sistem nonlinear yang memiliki banyak ketidakpastian. Ketidakpastian tersebut dapat berupa gangguan eksternal, ketidakpastian model, variasi parameter, ataupun *error* yang muncul pada saat linierisasi. Ketidakpastian-ketidakpastian ini dapat mempengaruhi kestabilan sistem jika tidak diantisipasi oleh sistem kontrol. Selain itu, pemodelan *Boiler* menggunakan konfigurasi sistem *Multi Input Multi Output* (MIMO) [1]. Sehingga variabel yang diatur pada *Boiler* mengalami kesulitan mengikuti sinyal referensi berupa beberapa kondisi titik operasi yang diinginkan [2].

Untuk mengatasi permasalahan tersebut dibuatkan model sistem kontrol menggunakan dekopling untuk mengatur konfigurasi plant MIMO dan kontrol PID untuk mengatur *level* dan *pressure Boiler-turbine.* Namun penentuan parameter dengan menggunakan metode *try and error* tidak efektif karena harus mencoba berulang kali untuk

mendapatkan hasil optimal. Maka pada penelitian ini digunakan metode PID dan Algoritma Genetika (GA) untuk *tuning* parameter dari kontroler. Kemudian plant dan sistem kontrol disimulasikan menggunakan *software* Wonderware.

1.2 Perumusan Masalah

Pemodelan *Boiler-turbine* menggunakan sistem *Multi Input Multi Output* (MIMO). Sehingga variabel yang diatur pada boiler mengalami kesulitan mengikuti sinyal referensi berupa kondisi titik yang diinginkan. Untuk mengatasi masalah tersebut, dirancang metode dekopling untuk mengatasi konfigurasi plant MIMO serta kontroler PID dan Algoritma Genetika sebagai *tuning* parameter untuk mengatur *level* dan *pressure* pada *Boiler-turbine*.

1.3 Batasan Masalah

Ruang lingkup pembahasan tugas akhir ini dibatasi sebagai berikut,

a. *Plant* yang dijadikan objek penelitian adalah unit *Boiler-turbine* Sydvenska Kraft AB di Malmo, Sweden dengan daya output maksimal sebesar 120MW.
b. *Plant* yang dikontrol *level* dan *pressure* pada simulator *Boiler*.
c. Kontroler yang digunakan adalah kontroler *proportional integral differential* (PID) dan metode dekopling yang digunakan untuk memisahkan interaksi antara kedua input plant boiler yang dibangun pada *software* MATLAB.
d. *Tuning* parameter kontroler PID menggunakan algoritma genetika.
e. *Tuning* parameter Kp, Ki, Kd dilakukan secara *offline*.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah merancang sistem kontrol menggunakan metode dekopling pada konfigurasi *plant* MIMO untuk menghilangkan interaksi *input feedwater* terhadap naik turunnya *pressure* dan perancangan kontroler PID dan GA sebagai *tuning* parameter untuk mengatur *level* dan *pressure Boiler-turbine* sehingga keluaran *level* dan *pressure* dapat diatur sesuai besar masukan referensi.

1.5 Sistematika Perancangan

Pembahasan Tugas Akhir ini akan dibagi menjadi lima bab sebagai berikut:

BAB 1 : PENDAHULUAN

Bab ini berisi latar belakang, perumusan masalah, tujuan penelitian, sistematika perancangan dan relevansi dari tugas akhir ini.

BAB 2 **:** TINJAUAN PUSTAKA

Berisi teori yang dijadikan acuan dalam penyusunan tugas akhir.

BAB 3 : PERANCANGAN SISTEM

Bab ini membahas proses perancangan simulator *Boiler-turbine* pada *software* MATLAB dan Wonderware

BAB 4 : PENGUJIAN DAN ANALISIS

Bab ini menjelaskan hasil simulasi dan analisa terhadap simulasi kontroler dengan dekopling pada Boiler-turbine

BAB 5 : KESIMPULAN DAN SARAN

Bab ini memaparkan kesimpulan dari yang diperoleh dari perancangan dekopling sistem MIMO dan kotroler PID untuk pengaturan *level* dan *pressure* pada *Boiler-turbine.*

1.6 Relevansi

Hasil yang diperoleh pada tugas akhir ini diharapkan dapat menjadi referensi perancangan boiler-turbine, pengembangan dan perbandingan metode kontrol yang tepat di masa mendatang.

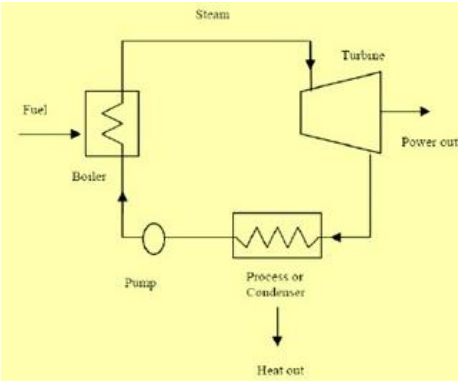
--halaman ini sengaja dikosongkan--

BAB 2
TEORI PENUNJANG

Bab ini membahas mengenai materi dasar dalam penyusunan Tugas Akhir. Beberapa hal yang akan dibahas meliputi tinjauan pustaka yang akan membahas mengenai sistem *Boiler-turbine*, *software* MATLAB 2015, Identifikasi Sistem, Dekopling MIMO, Kontroler *Proporsional Integral Derivative* (PID), Algoritma Genetika, dan *software* wonderware.

2.1 Boiler [1]

Boiler terdiri dari beberapa sistem, diantaranya adalah sistem *feedwater*, sistem *steam*, dan sistem *fuel* yang terintegrasi menjadi satu kesatuan. Sistem *feedwate*r berfungsi sebagai penyedia air untuk *Boiler* yang bekerja secara otomatis sesuai kebutuhan *steam* dan kemampuan dari *Boiler* itu sendiri. Sedangkan sistem *steam* berfungsi sebagai penyedia uap air untuk proses pada *plant* yang lain. Kedua sistem ini memerlukan suatu pengaturan agar kondisi air dan uap air dapat untuk memenuhi kebutuhan proses selanjutnya. Pengaturan air dilakukan dengan mengendalikan *level* air yang terdapat di dalam *Boiler* itu sendiri, sedangkan pengaturan uap air dilakukan dengan mengendalikan *pressure* dari *steam* yang merupakan hasil dari *Boiler* sendiri. *Pressure* yang dihasilkan oleh *Boiler* dipengaruhi oleh sistem *fuel* dan kondisi *level* air yang terdapat di dalam *Boiler*.



Gambar 2.1 Diagram sederhana *Boiler-turbine*

Pada Tugas Akhir ini, Model *boiler* yang digunakan merupakan sistem nonlinear yang telah dikembangkan dan diteliti oleh Bell dan Astrom pada tahun 1987. Parameter perhitungan yang digunakan pada *boiler-turbine* ini berasal dari data dinamis yang diukur dari plant Pembangkit Listrik Tenaga Uap (PLTU) yang berada di Sydvenska Kraft AB daerah dekat Malmo, Swedia. Pembangkit Listrik Tenaga Uap (PLTU) ini berbahan bakar minyak bumi dan dapat menghasilkan daya output maksimal sebesar 120 MW.

2.1.1 Model Matematika Boiler-Turbine

Persamaan diferensial pada boiler-turbine didefinisikan sebagai p, po, x_w , dimana :

p = tekanan drum (K_g/cm^2),

po = daya *output* (MW),

x_w = level drum (m^3)

Model matematika dalam persamaan diferensial dapat dituliskan sebagai berikut :

$$\frac{dP}{dt} = -0{,}0018u_2p^{\frac{9}{8}} + 0{,}9u_1 - 0{,}15u_3$$

$$\frac{dPo}{dt} = (0{,}073u_2 - 0{,}016)p^{\frac{9}{8}} - 0{,}1po$$

$$\frac{dPf}{dt} = \frac{(141u_3 - (1{,}1u_2 - 0{,}19)p}{85}$$

$$x_w = 0{,}05(0{,}23073_{pf} + 100 \propto_{cs} + \frac{q_e}{9-67{,}975}) \quad (2.1)$$

Dimana :

$$\propto_{cs} = \frac{(1-0{,}001538pf)(0{,}8p-25{,}6)}{pf(1{,}0394-0{,}0012304p)}$$

$$q_e = (0{,}854u_2 - 0{,}147)p + 45{,}59u_1 - 2{,}514u_3 - 2{,}096$$

Untuk input sistem (u_1, u_2, u_3) masing-masing merupakan posisi valve yang terdiri dari posisi valve untuk *fuel*, posisi valve untuk *steam*, dan control valve untuk aliran *feedwater*. Sedangkan $\propto_{cs}$ merupakan kualitas uap dan q_e sebagai laju penguapan.

Tetapi, pada Tugas Akhir ini, sistem yang dikendalikan ialah *pressure* dan *level.* Sehingga nilai steam dianggap konstan.

2.2 *Software* MATLAB 2015 [3]

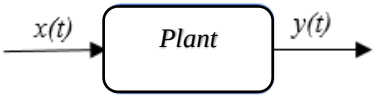
MATLAB merupakan paket program dengan bahasa pemrograman yang tinggi untuk mengembangkan algoritma, visualisasi data, dan komputasi numerik. Program MATLAB ini dapat digunakan untuk menyelesaikan masalah komputasi dengan lebih cepat dibandingkan dengan bahasa pemrograman tradisional, seperti C, C++, dan Fortran. MATLAB digunakan untuk banyak aplikasi seperti *signal and image processing*, desain kontrol, pengujian dan pengukuran, permodelan, dan analisis.

Simulink merupakan bagian dari MATLAB untuk memodelkan, mensimulasikan, dan menganalisa sistem dinamik. Simulink dapat membentuk model dari awal atau memodifikasi model yang sudah ada sesuai dengan apa yang diinginkan. Selain itu simulink juga mendukung sistem *linier* dan *non-linier*, permodelan waktu kontinyu atau diskrit, atau gabungan.

2.3 Identifikasi Sistem [4]

Identifikasi sistem dilakukan untuk mengetahui dinamika sistem dengan cara menyatakan dinamika sistem tersebut kedalam persamaan matematik. Persamaan matematik ini merupakan pendekatan yang representatif terhadap dari fenomena fisika dari sebuah sistem dinamik. Seluruh sistem dinamika di alam ini dapat dimodelkan dengan persamaan diferensial. Persamaan diferensial tersebut dapat diperoleh dengan menganalisa hukum-hukum fisis yang berlaku. Misalnya Hukum Newton untuk sistem mekanik, Hukum Kirch*off* untuk sistem elektrik.

Model matematik memiliki bentuk bermacam-macam tergantung sistem yang bersangkutan. Misalnya pada permasalahan analisa transien dari sebuah sistem LTI *single input single output (SISO)* akan lebih mudah jika direpresentasikan dengan *transfer function. Transfer function* merupakan sebuah persamaan matematika yang menggambarkan perbandingan dari fungsi *output* dalam *laplace* dan fungsi *input* dalam bentuk *laplace* .



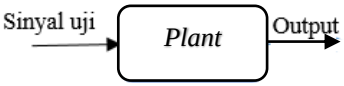
Gambar 2.2 Diagram Blok TF *Plant*

$$g(t) = \frac{y(t)}{x\,(t)} \tag{2.2}$$

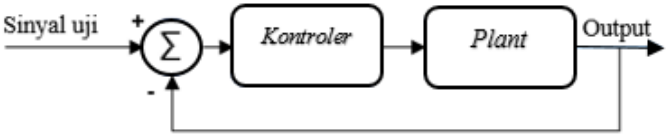
$$TF = G(s) = \frac{Y(s)}{X\,(s)} \tag{2.3}$$

Dimana g(t) merupakan model matematika dari *plant*, TF dapat dicari dengan melakukan transformasi *laplace* pada kedua sisi Persamaan (2.2) menjadi Persamaan (2.3). TF hanya digunakan untuk menyatakan sistem dengan *single input single output (SISO)* .

Teknik identifikasi ada dua macam, yaitu identifikasi *offline* dan *online*. Proses identifikasi *offline* dilakukan dengan cara mematikan kontroler dan memberikan sinyal uji kepada *plant* secara *open loop*, kemudian dinamika sinyal uji dan respon *plant* direkam untuk dianalisis lebih lanjut. Sedangkan proses identifikasi *online* dilakukan secara *close loop*. Artinya identifikasi *online* menggunakan kontroler. Proses identifikasi *offline* dan *online* dapat dilihat pada Gambar 2.3 dan Gambar 2.4.



Gambar 2.3 Proses Identifikasi *Offline*



Gambar 2.4 Proses Identifikasi *Online*

2.3.1 Identifikasi ARX

Secara umum, ada beberapa cara dalam mendapatkan model identifikasi. Salah satunya adalah identifikasi parameter. Identifikasi parameter sering digunakan untuk mencari model dengan respon transien yang tidak lazim. Identifikasi parameter dibagi menjadi 2, yaitu ARX dan ARMA.

ARX atau Auto-Regressive Exogenous input membentuk model matematis dengan menghasilkan model dalam domain diskrit dengan memiliki bagian numerator pada Persamaan (2.7) dan bagian denumerator seperti pada Persamaan (2.8):

$$A(q^{-1}) = a_0 + a_1 q^{-1} + \cdots + a_{nA} q^{-nA} \quad (2.7)$$

$$B(q^{-1}) = b_0 + b_1 q^{-1} + \cdots + b_{nB} q^{-nB} \quad (2.8)$$

Dengan $b_0 \neq 0;\ b_i = 0, i = 1, \ldots\ldots, n_B, a_j \neq 0;\ j = 1, \ldots\ldots, n_A$

2.3.2 Validasi

Tujuan validasi model adalah untuk menguji kesesuaian model hasil identifikasi dengan data respon plant. Untuk menguji kesesuaian model identifikasi dilakukan beberapa langkah, yaitu:

1. Penurunan loss function terhadap peningkatan jumlah parameter p. Kriteria ini berdasarkan pada nilai minimum dari Persamaan (2.9).

$$loss\ function = \frac{1}{2}\delta^2(N - p) \quad (2.9)$$

Dimana N menyatakan jumlah data hasil pengukuran. p menyatakan jumlah parameter model dan δ^2 merupakan variansi dari gangguan (residual).

2. Kriteria FPE (Final Prediction Error). Teknik ini berdasarkan nilai minimum dari suatu fungsi kriteria yang ditulis dalam Persamaan (2.10)

$$FPE = \frac{1+\frac{p}{N}}{1-\frac{p}{N}} V \quad (2.10)$$

Dengan V menyatakan loss function. Kesalahan prediksi rata-rata diharapkan mengecil jika jumlah dari parameter yang diestimasi meningkat.

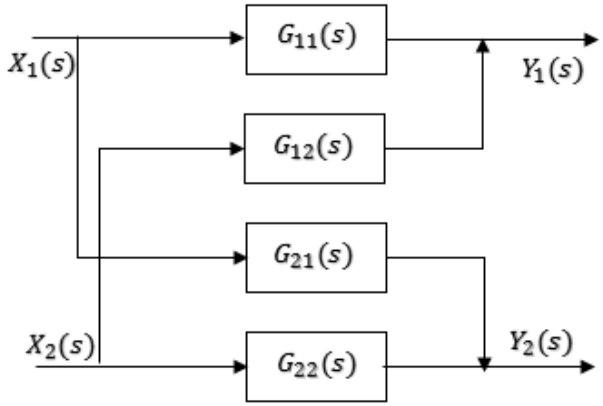
3. Kesalahan model dan analisi residual yang meliputi faktor ketidak bergantungan residual dan korelasi antara residual dengan masukan-keluaran.

4. Penurunan variasi residual estimasi terhadap peningkatan jumlah parameter.

5. Validasi silang dengan data yang belum pernah digunakan sebelumnya harus dapat membuktikan bahwa model estimasi mampu memprediksi perilaku sistem.

2.4 Dekopling MIMO [5]

Pada dunia industri, sering terjadi interaksi anatara variabel *input-output*. Perubahan suatu *input* kadang tidak hanya berpengaruh pada satu *output* saja, melainkan berpengaruh juga pada *output* lain, atau yang disebut dengan *Multi Input Multi Output* (MIMO). Pemodelan sistem ini dapat dilihat pada Gambar 2.5.



Gambar 2.5. *Plant* MIMO

Hubungan suatu *input* berpengaruh pada *output loop* lainnya. Penulisan model untuk Gambar 2.5. adalah

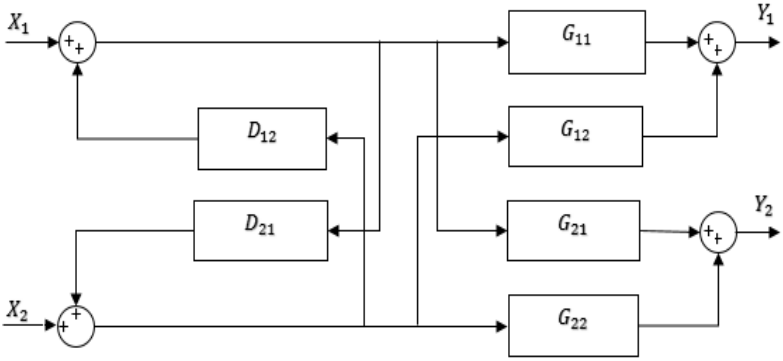
Loop 1 : $Y_1 = G_{11}X_1 + G_{12}X_2$ (2.11)

Loop 2 : $Y_2 = G_{21}X_1 + G_{22}X_2$ (2.12)

Untuk mengurangi interaksi *control loop*, dapat ditambahkan dekopler pada konfigurasi *multiloop* konvensional. Sistem kontrol dekopling memberi dua keuntungan:

a. Interaksi *control loop* i*nput* lain dihilangkan sehingga stabilitas sistem *closed loop* sistem ditentukan oleh karakteristik *feedback closed loop input* itu sendiri.
b. Perubahan *set point* pada satu pengubah terkendali tidak mempengaruhi pengubah-pengubah terkendali yang lain.

Salah satu jenis kontrol dekopling untuk proses dengan dua *input* dan dua *output* diperlihatkan pada Gambar 2.6. Tampak bahwa terdapat 2 dekopler D_{12} dan D_{21} yang berada sebelum diagram blok sistem MIMO.



Gambar 2.6. Desain Dekopling

Dekopler dirancang untuk mengkompensasi interaksi proses yang tidak diinginkan. Untuk memperoleh persamaan pada tiap dekopler maka dilakukan analisa terpisah pada tiap *input-output*, sehingga diperoleh nilai D_{12}, sebagai berikut:

$$Y_1 = G_{11}X_1^* + G_{12}X_2$$
$$X_1^* = X_1 + D_{12}X_2$$
$$Y_1 = G_{11}X_1 + G_{11}D_{12}X_2 + G_{12}X_2$$
$$Y_1 = G_{11}X_1 + (G_{11}D_{12} + G_{12})X_2 \quad (2.13)$$

Agar *output* Y_1 hanya dipengaruhi nilainya dari *input* X_1, nilai *input* X_2 harus sama dengan 0 (X_2= 0) Sehingga Persamaan menjadi:

$$(G_{11}D_{12} + G_{12})X_2 = 0$$
$$G_{11}D_{12} + G_{12} = 0$$
$$G_{11}D_{12} = -G_{12}$$
$$D_{12} = -\frac{G_{12}}{G_{11}} \quad (2.14)$$

Untuk nilai D1 diperoleh dari Persamaan berikut:

$$Y_2 = G_{22}X_2^* + G_{21}X_1$$
$$X_2^* = D_{21}X_1 + X_2$$
$$Y_2 = G_{22}D_{21}X_1 + G_{22}X_2 + G_{21}X_1$$
$$Y_2 = G_{22}X_2 + (G_{22}D_{21} + G_{21})X_1 \quad (2.15)$$

Agar *output* Y_2 hanya dipengaruhi nilainya dari *input* X_2, nilai X_1 harus sama dengan 0 (X_1=0), sehingga Persamaan menjadi:

$(G_{22}D_{21} + G_{21})X_1 = 0$

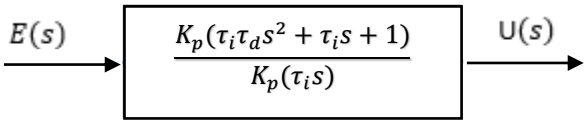
$(G_{22}D_{21} + G_{21}) = 0$

$$D_{21} = -\frac{G_{21}}{G_{22}} \quad (2.16)$$

2.5 Kontroler *Proporsional Integral Derivative* (PID) [6]

Kontroler merupakan salah satu komponen sistem yang berfungsi mengelolah sinyal umpan balik dan sinyal referensi menjadi sinyal kontrol sedemikian rupa, sehingga performasi dari sistem yang dikendalikannya sesuai dengan spesifikasi performansi yang diinginkan. Keberadaan kontroler dalam sebuah sistem kontrol mempunyai kontribusi besar terhadap perilaku sistem. Pada prinsipnya hal itu disebabkan oleh tidak dapat diubahnya komponen penyusun sistem tersebut. Artinya, karateristik plant harus diterima sebagaimana adanya, sehingga perubahan perilaku sistem hanya dapat dilakukan melalui penambahan suatu sub sistem, yaitu kontroler.

Kontroler PID merupakan penggabungan antara tiga macam kontroler, yaitu P (*Proportional*), I (*Integral*), dan D (*Derivative*). Setiap jenis kontroler memiliki karakteristik yang berbeda. Penggabungan kontroler dimaksudkan agar setiap kekurangan dan kelebihan dari masing – masing kontroler P, I, D dapat saling menutupi.



Gambar 2.7. Diagram blok kontroler standar PID

Bentuk standar dari kontroler PID dapat dituliskan seperti Persamaan (2.17) dimana u adalah sinyal kontrol, e adalah sinyal *error* ($e=y_{sp}-y$). Sinyal *error* adalah hasil penjumlahan dari tiga bagian, bagian P (nilainya proporsional terhadap sinyal *error*), bagian I (nilainya proporsional dan integral terhadap sinyal *error*), dan bagian D (nilainya proporsional dan derivatif terhadap sinyal *error*). Parameter dari kontroler adalah K_p, τ_i, dan τ_d.

$$u(t) = K_p[e(t) + \frac{1}{\tau i}\int e(t)dt + \tau_d \frac{d}{dt}e(t) \quad (2.17)$$

Saat sinyal kontrol yang diharapkan hanya aksi proporsional, maka bentuk Persamaan (2.17) direduksi menjadi Persamaan (2.18).

$$u(t) = K_P \, . e(t) \tag{2.18}$$

Kontroler proporsional menghasilkan *output* yang sebanding/proporsional dengan besarnya sinyal error. Yang secara sederhana dapat dikatakan, bahwa *output* kontroler proporsional merupakan perkalian antara konstanta proporsional dengan masukannya.

Fungsi dari aksi integral adalah agar *output* sistem sesuai dengan *set point*. Dengan aksi proporsional, akan muncul sinyal *error* pada saat *steady state*. Penggunaan aksi integral akan mengurangi sinyal *error* saat kondisi *steady state*. Struktur kontroler dengan aksi proporsional dan integral dinyatakan dalam Persamaan (2.19).

$$u(t) = K_p \left(e(t) + \frac{1}{\tau i} \int e(t) dt \right) \tag{2.19}$$

Aksi derivatif diperlukan untuk meningkatkan stabilitas *closed loop*. Saat ada perubahan variabel kontrol, dibutuhkan waktu untuk mengoreksi *error* sehingga ada keterlambatan pada sinyal kontrol. Aksi kontroler dengan proporsional dan derivatif bernilai proporsional terhadap *output* yang diprediksi. Struktur kontroler dengan aksi proporsional dan derivatif dinyatakan dalam Persamaan (2.20).

$$u(t) = K_p \left(e(t) + \tau_d \frac{d}{dt} e(t) \right) \tag{2.20}$$

Tabel 2.1 Pengaruh K_p, K_i, K_d pada respon sistem

Respon *Closed-Loop*	*Rise Time*	*Overshoot*	*Settling Time*	*SS Errror*
K_p	Turun	Naik	Perubahan Kecil	Turun
K_i	Turun	Naik	Naik	Hilang
K_d	Perubahan Kecil	Turun	Turun	Perubahan Kecil

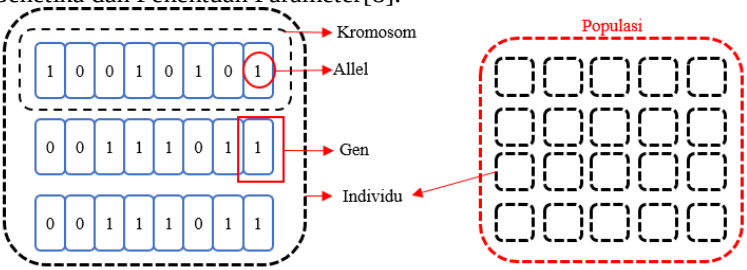
2.6 Algoritma Genetika [7]

Algoritma genetika sebagai cabang dari algoritma evolusi merupakan metode *adaptive* yang bisa memecahkan suatu pencarian nilai dalam sebuah masalah optimasi. Algoritma ini bekerja dengan sebuah populasi yang terdiri dari individu-individu, yang masing-masing individu dilambangkan dengan sebuah nilai *fitness* yang akan digunakan untuk mencari solusi terbaik dari permasalahan.

Dengan teori evolusi dan teori genetika, pada algoritma genetika akan melibatkan beberapa proses. Yaitu: proses seleksi, proses pindah silang (*crossover)*, dan mutasi.

2.6.1 Komponen Komponen Utama Algoritma Genetika

Ada enam komponen utam dalam algoritma genetika, yaitu teknik penyandian, Prosedur Inisialisasi, fungsi Evaluasi, Seleksi, Operator Genetika dan Penentuan Parameter[8].



Gambar 2.8. Istilah dalam Algoritma Genetika

Terdapat juga beberapa definisi penting dalam algoritma genetika, antara lain:

a. *Genotype* (Gen), sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom.
b. Allel, nilai dari gen.
c. Kromosom, gabungan gen-gen yang membentuk nilai tertentu.
d. Individu, menyatakan suatu nilai yang menyatakan salah satu solusi yang mungkin untuk suatu permasalahan optimasi.
e. Populasi, Kumpulan individu yang akan diproses dalam satu siklus algoritma genetika.

f. Generasi, menyatakan jumlah siklus algoritma genetika berjalan.

2.6.2 Teknik Penyandian

Teknik penyandian di sini meliputi penyandian gen dan kromosom. Gen merupakan bagian dari kromosom. Satu gen biasanya akan mewakili satu variabel. Gen dapat direpresentasikan dalam bentuk *string bit*, pohon, *array* bilangan *real*, daftar aturan, elemen permutasi, elemen program, atau representasi lainnya yang dapat diimplementasikan untuk operator genetika.

2.6.3 Prosedur Inisialisasi

Ukuran populasi tergantung pada masalah yang akan dipecahkan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian harus dilakukan inisialisi terhadap kromosom yang terdapat pada populasi tersebut. Inisialisasi kromosom dilakukan secara acak, namun demikian harus tetap memperhatikan domain solusi dan kendala permasalahan yang ada.

2.6.4 Fungsi Evaluasi

Ada dua hal yang harus dilakukan dalam melakukan evaluasi kromosom, yaitu evaluasi fungsi objektif (fungsi tujuan) dan konversi fungsi objektif kedalam fungsi *fitness*. Secara umum, fungsi *fitness* diturunkan dari fungsi objektif yang tidak negatif.

2.6.5 Seleksi

Seleksi ini bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi individu yang *fitness* nya lebih besar. Seleksi akan menentukan individu-individu mana saja yang akan dipilih untuk dilakukan rekombinasi dan bagaimana *offspring* terentuk dari individu-individu terpilih tersebut. Ada beberapa metode seleksi dari induk, antara lain *Rank-based fitness assigment, Roulette wheel selection, Stochastic universal sampling, Local selection, Truncation selection, Tournament selection.*

2.6.6 Operator Genetika

Secara umum, operasi pada algoritma genetika ada dua, yaitu *crossover* dan mutasi. Jenis *crossover* antara lain:

a. Crossover satu titik.
b. Crossover banyak titik.

c. *Crossover uniform.*
d. *Crossover* dengan permutasi.

Setelah mengalami proses rekombinasi, pada *offspring* dapat dilakukan mutasi. Variabel *offspring* dimutasi dengan menambahkan nilai random yang sangat kecil (ukuran langkah mutasi), dengan probabilitas yang rendah. Mutasi berperan untuk menggantikan gen yang hilang dari populasi akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi. Selain itu mutasi juga berperan untuk memperlebar ruang pencarian. Mutasi dapat dibagi menjadi dua yaitu mutasi bernilai real dan mutasi bernilai biner.

2.6.7 Kondisi *Stopping*

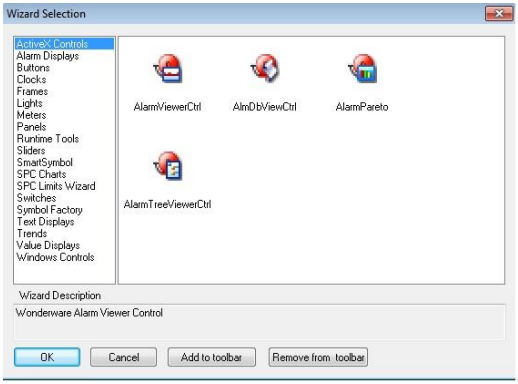
Proses *stopping* bisa menggunakan beberapa kriteria. Antara lain jumlah generasi, lama waktu program berjalan, tercapainya nilai *fitness* tertentu, dan saat *fitness* tidak ada perubahan.

2.7 Software Wonderware Intouch [9]

Software Wonderware Intouch merupakan *software* yang dikembangkan untuk HMI pada bidang Industri. Wonderware Intouch dapat digunakan dalam mode *online* dan *offline*. Pada mode *online* masukan dan keluaran terhubung seperti contohnya *PLC (Programmable Logic Controller)*. Sedangkan pada mode *offline* program dijalankan dengan fasilitas pada *software* tersebut.

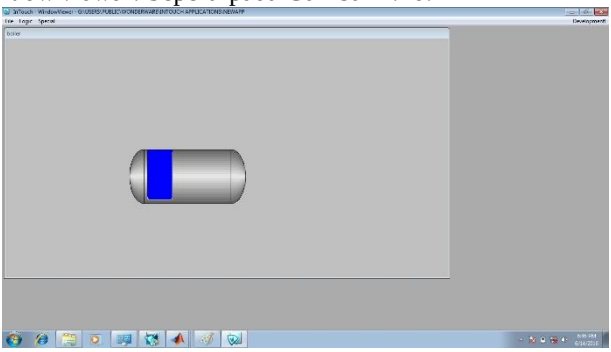
Intouch berisi tiga komponen utama, yaitu *Application Manager*, *Window Maker*, dan *Window Viewer*. Fungsi *Application manager* ialah untuk mengatur aplikasi yang akan dibuat. Sedang pada *window Maker* digunakan untuk membuat aplikasi-aplikasi yang akan didesain. *Window Viewer* berupa *runtime enviroment* yang digunakan untuk menampilkan grafik yang dibuat di *window maker*.

Dalam *window maker*, tiap variabel harus di *assigned* sebuah *tagname* dan tipenya. Tipe tagname dapat berupa *memory tagname* atau I/O *tagname*. Cara untuk membuat tampilan grafik atau animasi dilakukan dengan mengetag komponen-komponen sesuai keperluan desain. Semua kompone tersebut tersedia di dalam *feature wizard selection* seperti tampak pada Gambar 2.9.



Gambar 2.9. Tampilan Wizard Selection

Setelah fungsi objek yang berada pada *window maker* didefinisikan, maka selanjutnya adalah menuliskan program pada *script.* Jika ingin melihat hasil dari desain simulasi yang telah dibuat, maka dapat dilihat pada *window viewer*. Seperti pada Gambar 2.10.



Gambar 2.10. Tampilan Window Viewer

Untuk membuka *window viewer* dengan mengklik tombol *runtime* pada *window maker* .

2.8 OPC KEPServerEx 5 [9]

Perangkat lunak yang digunakan dalam penelitian adalah Wonderware dan MATLAB. Untuk menghubungkan keduanya, dibutuhkan protokol komunikasi yang menjebatani keduanya. Protokol

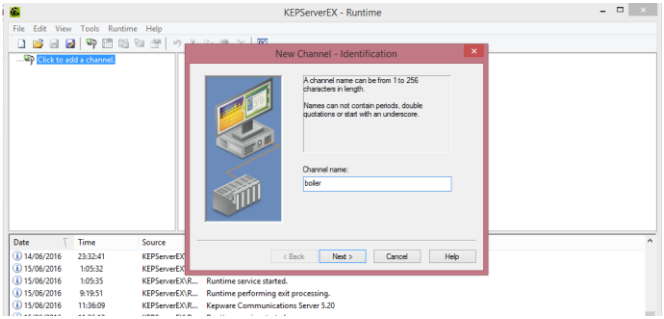
komunikasi yang menjebatani adalah OPC (OLE for process Kontrol.). OPC yang digunakan ialah OPC KEPServerEx.

KEPServerEx adalah generasi terbaru teknologi server OPC dari Kepware yang mendukung teknologi client server berikut :

a. Akses Data OPC versi 1.0a
b. OPC Data Access Versi 2.0
c. FastDDE untuk Wonderware
d. SuiteLink untuk Wonderware
e. DDE Format CF_Text
f. DDE Format Advanced DDE

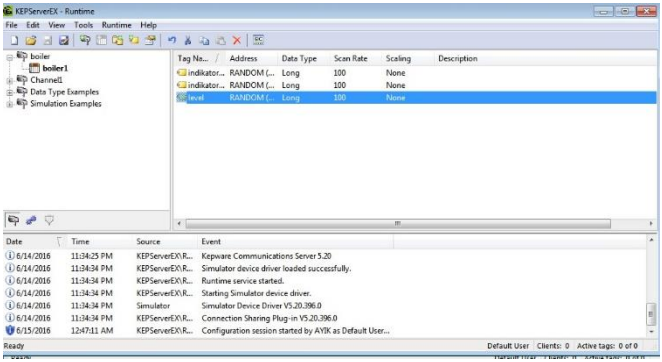
KEPServerEx adalah solusi yang fleksibel dan scalable untuk menghubungkan, mengelola, memantau, dan mengendalikan perangkat otomatisasi yang beragam dan aplikasi perangkat lunak.

KEPServerEx mendukung penggunaan modem pada semua *driver* komunikasi serial. Kontrol modem disediakan oleh satu set tag modem baru. Setelah operasi modem diaktifkan untuk proyek KEPServerEx, satu set standar tag modem menjadi tersedia untuk aplikasi *client.*



Gambar 2.11. Tampilan OPC KEPServerEx 5

Pada saat membuat sebuah *project* Intouch, intouch berperan sebagai *client* pada KEPServerEx 5. Langkah awalnya dengan membuat aplikasi baru pada kotak dialog dan masukkan nama *channel* dan pilih intouch *client driver* pada *device driver* seperti Gambar 2.11.



Gambar 2.12. Tampilan *Tagname* pada KEPServerEx 5

Untuk mengoneksikan KEPServerEx dengan Wonderware, maka pada wonderware harus dibuat *access name* yang akan mengakses *channel* dan *device* pada KEPServerEx yang akan dibaca datanya. Lalu pada item name isikan dengan tag pada KEPServerEX 5 yang akan dibaca datanya seperti pada Gambar 2.12. Setelah itu KEPServerEx 5 dan *Runtime* aplikasi pada Wonderware Intouch dapat dijalankan.

Halaman ini sengaja dikosongkan

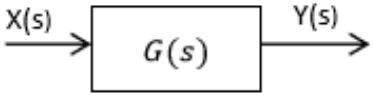
BAB 3
PERANCANGAN SISTEM

Pada Bab ini membahas mengenai metodelogi penelitian yang digunakan. *Plant* yang digunakan adalah *boiler-turbine*. Diawali dengan identifikasi sistem untuk menentukan model matematika sistem pada sistem linear menggunakan pendekatan parametrik ARX. Perancangan dekopling untuk menghilangkan sifat mempengaruhi interaksi *input-output*. Setelah terpasang dekopler, *plant* asli berupa sistem nonlinear dipasang lalu merancang kontroler PID dengan *tuning* menggunakan Algoritma Genetika (GA). Kemudian simulasi dilakukan pada diagram blok simulink MATLAB dengan wonderware yang digunakan untuk visualisasi *plant.*

3.1 Identifikasi Sistem

Identifikasi sistem diperlukan untuk mendapatkan model matematika dari sistem. Pada bagian ini akan dibahas bagaimana mendapatkan identifikasi dinamis bentuk ARX serta transformasi model dari domain waktu disktit ke domain Z lalu ke domain *Laplace.*

3.1.1 Identifikasi Sistem ARX



Gambar 3.1. Identifikasi Sistem

Proses identifikasi sistem menggunakan cara identifikasi dinamis bentuk ARX (*Auto Regressive Exogeneous*) dengan validasi *error* menggunakan metode FPE (*Final Prediction Error*) pada MATLAB. Sehingga akan mendapatkan *transfer fucntion* dari sistem seperti pada Gambar 3.1.

Model yang didapat dalam bentuk ARX dengan FPE terkecil dengan domain diskrit ditransformasi ke domain Z, setelah itu model yang didapatkan ditransformasi kembali ke domain *Laplace* sehingga didapatkan model sebagai berikut.

Identifikasi model untuk u_1 pada *output pressure*:

$$y(k) = -0.06199y(k-1) - 0.0134y(k-2) + 131x(k-1) + 0.004795x(k-2) \quad (3.1)$$

Persamaan (3.1) hasil transformasi transformasi ke domain Z dapat ditulis dalam bentuk:

$$\frac{Y(z)}{X(z)} = \frac{131z+0.004795}{z^2-0.6199z+0.0134} \quad (3.2)$$

Persamaan (3.2) hasil transformasi transformasi ke domain *Laplace* dapat ditulis dalam bentuk:

$$\frac{Y(s)}{X(s)} = \frac{0.7714s+0.01628}{s^2+0.02156s+0.0000489} \quad (3.3)$$

Identifikasi model untuk u_1 pada *output level*:

$$y(k) = -0.002789y(k-1) - 0.005338y(k-2) + 55.87x(k-1) + 55.85x(k-2) \quad (3.4)$$

Persamaan (3.4) hasil transformasi transformasi ke domain Z dapat ditulis dalam bentuk:

$$\frac{Y(z)}{X(z)} = \frac{55.87z+55.85}{z^2+0.002789z+0.005338} \quad (3.5)$$

Persamaan (3.5) hasil transformasi transformasi ke domain *Laplace* dapat ditulis dalam bentuk:

$$\frac{Y(s)}{X(s)} = \frac{-4.429s+0.02578}{s^2+0.02616s+0.0002313} \quad (3.6)$$

Identifikasi model untuk u_3 pada *output pressur*e:

$$y(k) = -0.6545y(k-1) + 0.01346y(k-2) + 41.06x(k-1) + 0.01616x(k-2) \quad (3.7)$$

Persamaan (3.7) hasil transformasi transformasi ke domain Z dapat ditulis dalam bentuk:

$$\frac{Y(z)}{X(z)} = \frac{41.06z+0.01616}{z^2-0.6545z+0.01346} \tag{3.8}$$

Persamaan (3.8) hasil transformasi transformasi ke domain *Laplace* dapat ditulis dalam bentuk:

$$\frac{Y(s)}{X(s)} = \frac{0.2362s+0.005034}{s^2+0.02154s+0.00004399} \tag{3.9}$$

Identifikasi model untuk u_3 pada *output level*:

$$y(k) = -0.0001953y(k-1) - 0.000158y(k-2) + 2.508x(k-1) + 2.508x(k-2) \tag{3.10}$$

Persamaan (3.10) hasil transformasi ke domain Z dapat ditulis dalam bentuk:

$$\frac{Y(z)}{X(z)} = \frac{2.508z+2.508}{z^2-0.0001953z+0.000158} \tag{3.11}$$

Persamaan (3.11) hasil transformasi ke domain *Laplace* dapat ditulis dalam bentuk:

$$\frac{Y(s)}{X(s)} = \frac{-1.449s+0.002708}{s^2+0.04376s+0.0005399} \tag{3.12}$$

3.2 Transfer Function *Boiler-Turbine*

Bentuk *transfer function boiler-turbine* didapat dari hasil identifikasi sistem sebelumnya, yaitu:

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix} \tag{3.13}$$

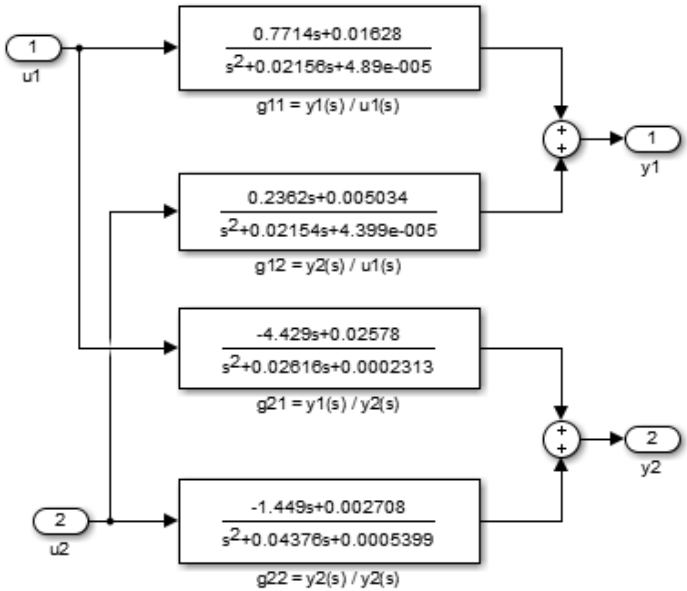
Dengan u_1, u_2, y_1, y_2 secara berturut-turut merupakan *fuel*, *feedwater*,*pressure, d*an *level.* Maka didapat:

$$G_{11}(s) = \frac{0.7714s+0.01628}{s^2+0.02156s+0.0000489} \quad (3.14)$$

$$G_{12}(s) = \frac{0.2362s+0.005034}{s^2+0.02154s+0.0000399} \quad (3.15)$$

$$G_{21}(s) = \frac{-4.429s+0.02578}{s^2+0.02616s+0.0002313} \quad (3.16)$$

$$G_{22}(s) = \frac{-1.449s+0.002708}{s^2+0.04376s+0.0005399} \quad (3.17)$$



Gambar 3.2. *Transfer Function Boiler-Turbine*

Gambar 3.2 merupakan bentuk konfiguransi MIMO untuk transfer function dari *boiler-turbine.* Persamaan (3.14) sampai Persamaan (3.17)

merupakan *transfer function boiler-turbine* dari identifikasi sistem pada Bab 3 Bagian 3.1.

3.3 Desain Decoupling

Pada sistem MIMO *(Multi Input Multi Output)* masing-masing input mempengaruhi kedua *output*. Oleh karena itu, dirancang suatu metode dekopling untuk menghilangkan pengaruh interaksi *input-output*. Tetapi, pada tugas Akhir kali ini, metode decoupling hanya digunakan pada *pressure* sehingga i*nput* hanya dipengaruhi oleh *fuel*. Sedangkan pada *level* tidak dipasang decoupler karena sistem susah untuk dikendalikan. Sehingga naik turun level dipengaruhi oleh *fuel* dan *feedwater*.

Persamaan dekopling didapat dengan memasukkan nilai matriks *transfer function boiler-turbine* yang telah didapatkan dari Persamaan (3.14) sampai Persamaan (3.17) didapatkan persamaan matriks dekopling sebagai berikut:

$$D(s) = \begin{bmatrix} 1 & \frac{-G_{12}(s)}{G_{11}(s)} \\ \frac{-G_{21}(s)}{G_{22}(s)} & 1 \end{bmatrix} \tag{3.18}$$

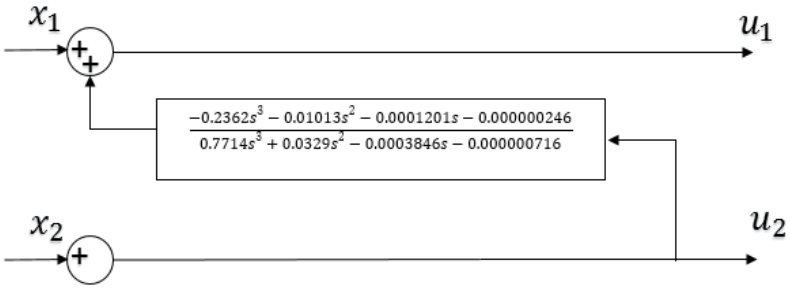
Dengan hanya menggunakan *satu* dekopling untuk *pressure* sebagai *output* y1 maka matriks yang didapat dekopling *turbine boiler:*

$$D(s) = \begin{bmatrix} 1 & \frac{-G_{12}(s)}{G_{11}(s)} \\ 1 & 1 \end{bmatrix} \tag{3.19}$$

Dengan *transfer function* yang didapat :

$$D_{12}(s) = \frac{-0.2362s^3 - 0.01013s^2 - 0.0001201s - 0.000000246}{0.7714s^3 + 0.0329s^2 - 0.0003846s - 0.000000716} \tag{3.20}$$

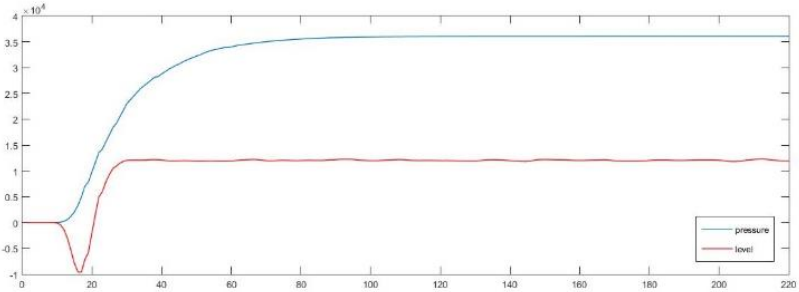
Gambar dekopling sistem pada penelitian ini ditunjukan pada Gambar 3.3. Dengan adanya metode dekopling pada sistem, maka *output* Y_1 hanya dipengaruhi oleh *input* u_1. Sedangkan *output* Y_2 dipengaruhi oleh i*nput* u_1 dan u_2.



Gambar 3.3. Desain Dekopling *Boiler-Turbine*

Setelah diperoleh nilai dekopler D_{12} maka perlu diuji apakah dekopler sudah dapat menghilangkan interaksi antar *input*. Pengujian dilakukan dengan bantuan simulink MATLAB R2015a.

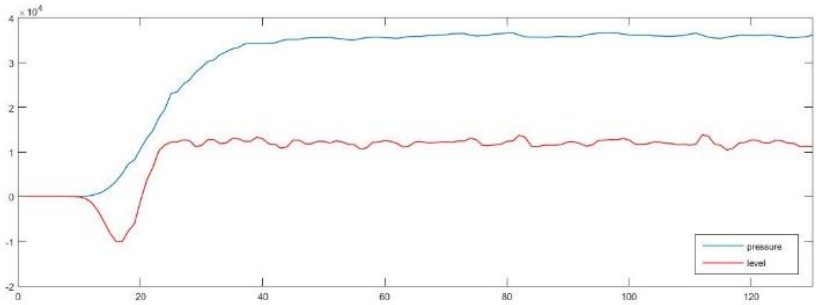
Pengujian dekopling dilakukan dengan memberi sinyal random 10% dari *input* u_2. Hasil respon pada sistem saat *input* u_2 diberi gangguan ditunjukan pada Gambar 3.4.



Gambar 3.4. Respon ketika *Input* u_2 Diberi Gangguan

Dari Gambar 3.4 terlihat apabila gangguan diberikan pada sistem *level*, maka *output* yang terpengaruh hanya pada sistem *level*, sedangkan pada sistem *pressure* tidak terpengaruh dari gangguan. Sedangkan ketika pengujian dekopling dilakukan dengan memberi sinyal random 10% dari *input* u_1. Hasil respon pada sistem *level* ikut terpengaruh. Hal itu ditunjukan pada Gambar 3.5.

Dari Gambar 3.5 terlihat apabila gangguan diberikan pada i*nput* u_1, *sistem level* dan *pressure* juga mengalahi perubahan.



Gambar 3.5. Respon ketika *Input* u_1 Diberi Gangguan

3.4 Perancangan Kontroler

Setelah didapat dekopler untuk sistem *pressure* maka *plant* dikembalikan pada sistem nonlinear untuk selanjutnya dirancang kontroler. Kontroler digunakan untuk mengontrol *pressure* dan *level* pada *Boiler*. Tahapan desain kontroler ini meliputi perancangan kontroler PID dan perancangan algoritma genetika.

3.4.1 Kontroler PID

PID digunakan untuk mengontrol *pressure* dan *level Boiler* dengan menggunakan struktur paralel seperti pada Persamaan (3.21) Parameter Kp, Ki, Kd akan dituning menggunakan algoritma genetika.

$$C(s) = K_p + \frac{K_i}{s} + K_d s \tag{3.21}$$

Objektif dari kontroler PID adalah menghasilkan respon output menyerupai sistem nonlinear dengan *steady error = 0%* dan *max error overshoot = 6%*.

3.4.2 Algoritma Genetika

Algoritma genetika yang digunakan pada penelitian ini digunakan untuk melakukan tuning parameter dari kontroler PID. Algoritma genetika didesain menggunakan software Matlab 2015.

Individu pada algoritma genetika berisi enam gen yang menyatakan nilai Kp, Ki, Kd masing-masing untuk *pressure* dan *level*. Proses

pencarian nilai parameter dari setiap individu digunakan fungsi *fitness*. Penilaian optimal atau tidaknya suatu individu berdasarkan pengujiannya menggunakan fungsi *fitness.* Parameter fungsi *fitness* terdiri dari *overshoot* dan *relative steady state error*. Hasil akhir dari algoritma genetika diharapkan didapatkan individu dengan nilai *fitness* paling besar yang mereperesentasikan solusi terbaik dari parameter kontroler PID.

$K_p p$	$K_i p$	$K_d p$	$K_p l$	$K_i l$	$K_i l$

Gambar 3.6 Individu pada Algoritma Genetika

Langkah pertama dari algoritma genetika adalah membangkitkan populasi dan generasi. Pembangkitan nilai gen dari individu dilakukan secara random dengan range kerja tiap gen adalah sebagai berikut:

a. Pp = 0 - 0146
b. Ip = 0 – 0.0001
c. Dp = 0 – 0.00000000001
d. Pl = 0 – 0.2663
e. Il = 0 – 0.000000001
f. Dl = 0 – 0.00000000000001

Selanjutnya dilakukan perhitungan nilai *fitness* dengan menggunakan populasi yang telah dibangkitkan sehingga setiap individu memiliki nilai *fitness* yang berbeda-beda. Fungsi *fitness* dinyatakan dalam persamaan:

$$f(i) = RMSE_{(P)} + RMSE_{(l)} + os_{(p)} + os_{(l)}$$
$$k(u,1) = 20.000/f(i) \quad (3.22)$$

$RMSE_{(P)}$ = *root mean-square error* pada *pressure*
$RMSE_{(l)}$ = *root mean-square error* pada *level*
$os_{(p)}$ = *overshoot* pada *pressure*
$os_{(l)}$ = *overshoot* pada *level*

Dengan penghitungan nilai RMSE dan overshoot terdapat pada Persamaan (3.23) dan Persamaan (3.24):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - Y_i)^2} \quad (3.23)$$

$Overshoot = \sum_{i=1}^{n} \max Y_n - X_i$ (3.24)

Setelah nilai *fitness* dihitung, individu diurutkan berdasarkan nilai *fitness*nya. Langkah selanjutnya adalah seleksi. Seleksi yang digunakan ialah *trucking* dan *roullete.* Pada seleksi *trucking*, individu dengan nilai *fitness* yang tinggi yang akan terpilih menjadi induk. Banyaknya individu yang terpilih bergantung pada rasio nilai seleksi. Individu yang hilang dari proses seleksi *truncking* akan dikembalikan melalui seleksi *roullete.*

Lalu, tahap selanjutnya adalah *crossover*. Metode yang digunakan ialah dengan membangkitkan nilai random integer yang merepresentasikan nilai *cut point.* Individu baru tersebut tergantung pada rasio *crossover.*

Jika *cut point* 1 yang terpilih, maka individu baru tersebut akan memiliki persamaan:

$x1 = a\ x\ Rc + g\ x\ (1 - Rc)$ (3.25)

$x2$=b, $x3$=c, $x4$=d, $x5$=e, $x6$=f (3.26)

Jika *cut point* 2 yang terpilih, maka individu baru tersebut akan memiliki persamaan:

$x1 = a\ x\ Rc + g\ x\ (1 - Rc)$ (3.27)

$x2 = b\ x\ Rc + h\ x\ (1 - Rc)$ (3.28)

$\boldsymbol{x3}$**=c,** $\boldsymbol{x4}$**=**d, $x5$=e, $x6$=f (3.29)

Jika *cut point* 3 yang terpilih, maka individu baru tersebut akan memiliki persamaan:

$x1 = a\ x\ Rc + g\ x\ (1 - Rc)$ (3.30)

$x2 = b\ x\ Rc + h\ x\ (1 - Rc)$ (3.31)

$x3 = c\ x\ Rc + i\ x\ (1 - Rc)$ (3.32)

$x4$=d, $x5$=e, $x6$=f (3.33)

Jika *cut point* 4 yang terpilih, maka individu baru tersebut akan memiliki persamaan:

$x1 = a\ x\ Rc + g\ x\ (1 - Rc)$ (3.34)

$x2 = b\ x\ Rc + h\ x\ (1 - Rc)$ (3.35)

$x3 = c\ x\ Rc + i\ x\ (1 - Rc)$ (3.36)

$$x4 = d\ x\ Rc + j\ x\ (1 - Rc) \quad (3.37)$$
$$x5 = e, x6 = f \quad (3.38)$$

Jika *cut point* 5 yang terpilih, maka individu baru tersebut akan memiliki persamaan:

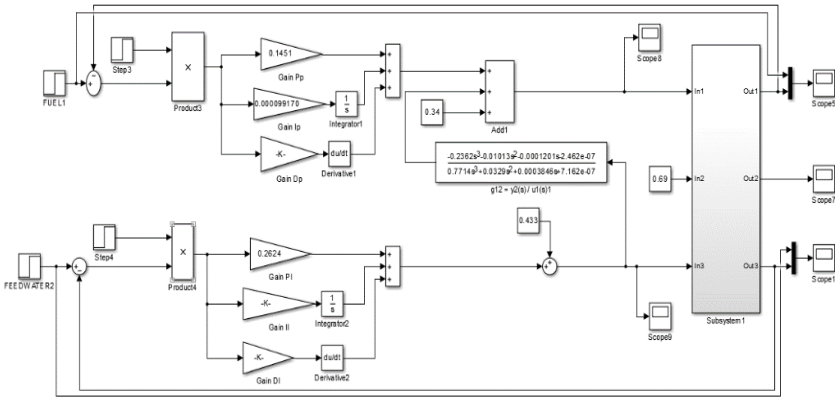
$$x1 = a\ x\ Rc + g\ x\ (1 - Rc) \quad (3.39)$$
$$x2 = b\ x\ Rc + h\ x\ (1 - Rc) \quad (3.40)$$
$$x3 = c\ x\ Rc + i\ x\ (1 - Rc) \quad (3.41)$$
$$x4 = d\ x\ Rc + j\ x\ (1 - Rc) \quad (3.42)$$
$$x5 = e\ x\ Rc + k\ x\ (1 - Rc) \quad (3.43)$$
$$x6 = f \quad (3.44)$$

Setelah *crossover*, langkah selanjutnya adalah mutasi. Banyak gen yang mengalami mutasi tergantung pada rasio mutasi. Untuk memilih posisi gen yang mengalami mutasi, dibangkitkan nilai random dari banyaknya gen yang mengalami mutasi. Setelah mengalami mutasi, dan kondisi *stopping* belum terpenuhi maka proses selanjutnya menghitung nilai *fitness* kembali hingga memenuhi generasi yang telah ditentukan.

3.5 Perancangan Simulasi Boiler-turbine

Hasil perancangan *Boiler-Turbine*, konfigurasi dekopling, dan kontroler disimulasikan pada simulink MATLAB. Pada simulink terdapat blok diagram plant nonlinear *Boiler-Turbine*, dekopling agar *output pressure* tidak dipengaruhi oleh *input*. Lalu ada dua kontroler PID yang masing-masing bertujuan untuk mengendalikan nilai *pressure* dan *level* agar sama dengan sinyal referensi. Selain itu terdapat *operating point* pada *pressure, level*, dan *steam*, dimana sinyal kontrol berjalan pada titik kerja yang telah ditentukan.

Diagram simulink sistem kontrol *Boiler-turbine* secara keseluruhan ditunjukkan dengan Gambar 3.7.

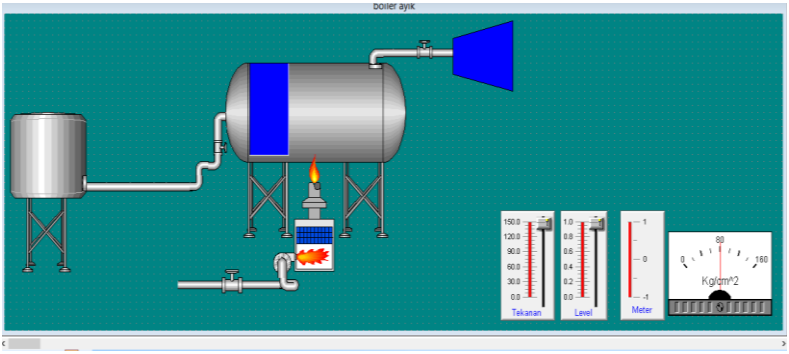


Gambar 3.7 Diagram Simulink *Boiler-Turbine* Secara Keseluruhan

3.6 Perancangan HMI

Virtual merupakan salah satu sarana penghubung antara manusia dengan mesin. Output sistem perlu ditampilkan kepada pengguna secara visualisasi. Oleh karena itu dirancang sebuah virtual *boiler-turbine* melalui *software wonderware*.

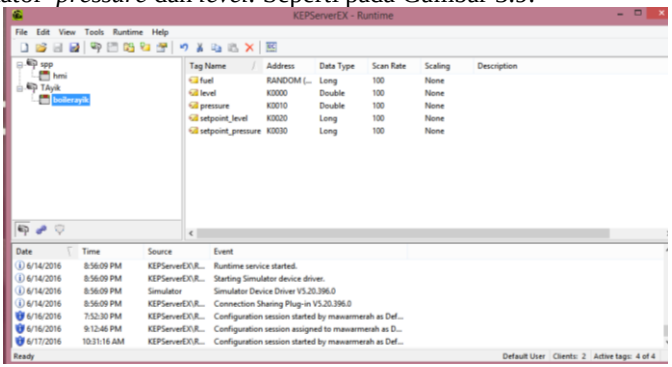
Pemodelan *plant*, dekopling dan kontroler plant dibangun dalam simulink MATLAB yang dihubungkan pada wonderware menggunakan KEPserverEx.



Gambar 3.8 Tampilan *Human Mechine Interface*

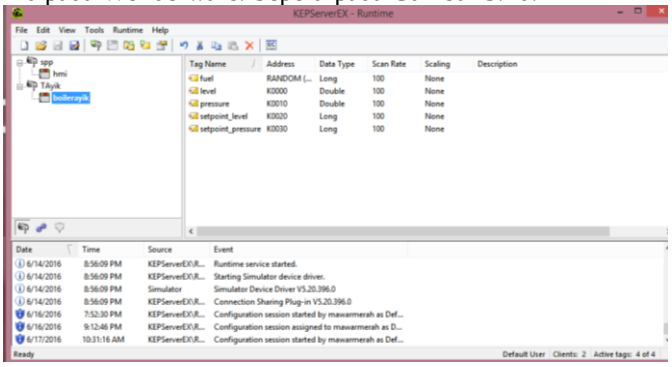
Untuk membuat virtual *boiler-turbine*, *p*ertama dibangun sebuah *Boiler-turbine* dengan tiga *valve* yang masing-masing merupakan *valve feedwater*, *valve fuel*, dan *valve steam* dengan indikator dua *setpoint*, *pressure* dan *level.*

Lalu, untuk memberikan animasi, diberikan tagname untuk air, indikator *pressure* dan *level*. Seperti pada Gambar 3.9.



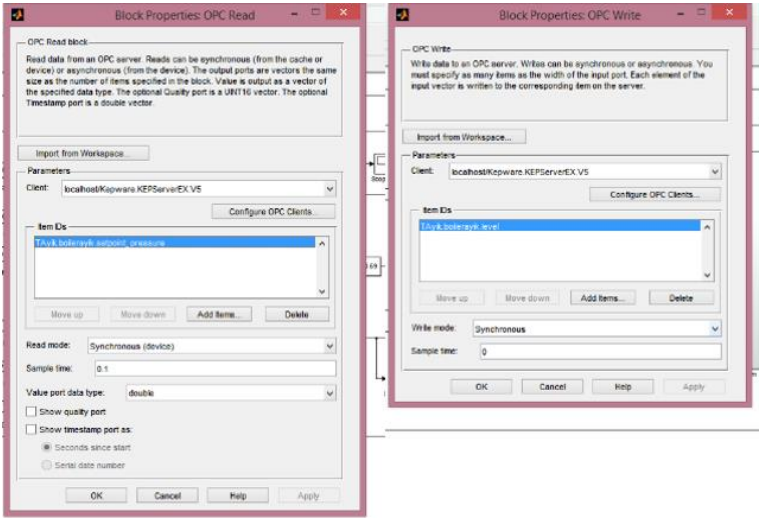
Gambar 3.9. Pemberian *Tagname* untuk HMI *Plant Boiler-Turbine*

Setelah pemberian *tagname* keseluruh bagian *virtual plaint Boiler-Turbine* yang akan diberi animasi. Selanjutnya, mengkomunikasikan simulasi MATLAB pada Wonderware menggunakan KEPServerEx 5. Pada KEPServerEx 5, MATLAB dan Wonderware merupakan *client* pada KEPServerEx 5. Langkah pertama adalah membuat *channel* pada KEPServerEx 5, lalu beri *tagname* pada *channel* tersebut sesuai dengan *tagname* pada Wonderware. Seperti pada Gambar 3.10.



Gambar 3.10. Tampilan *Tagname* pada KEPServerEx 5

Sedangkan pada simulasi MATLAB, dengan bantuan toolbox OPC MATLAB (OPC read dan OPC write) diatur konfigurasi sesuai dengan KEPServerEx5 seperti pada gambar 3.11.



Gambar 3.11. Tampilan Konfigurasi OPC *Read* dan OPC *Write* pada MATLAB

Halaman ini sengaja dikosongkan

BAB 4
PENGUJIAN DAN ANALISIS

Pada bab ini dibahas mengenai simulasi dari hasil perancangan yang telah dibuat pada Bab 3. Setelah dilakukan simulasi kemudian dilakukan analisa terhadap data hasil simulasi yang terbagi menjadi simulasi kontroler PID-GA *Boiler-Turbine* menggunakan dekopling, Uji beban, dan Tampilan HMI saat disimulasikan.

4.1 Simulasi Kontroler PID-GA *Boiler-turbine* Menggunakan Dekopling

Proses *tuning* parameter kontroler PID dilakukan pada kondisi beban nominal menggunakan algoritma genetika. Selama proses *tuning*, parameter algoritma genetika yang diubah-ubah adalah populasi (P), generasi (G), rasio seleksi (Rs), rasio *crossover* (Rc), dan rasio mutasi (Rm).

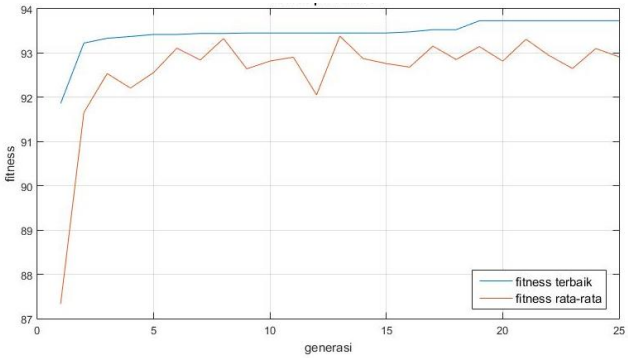
Pada pegujian Tugas Akhir ini dilakukan lima kali percobaan dengan merubah rasio seleksi (Rs), rasio *crossover* (Rc), dan rasio mutasi (Rm) sebanding secara berturut-turut senilai: 0,1, 0,2, 0,3, 0,4, 0,5 dengan jumlah populasi dan jumlah generasi sama yaitu sejumlah 25.

Pada pengujian pertama penulis menggunakan parameter algoritma genetika yang terdapat pada Tabel 4.1.

Tabel 4.1 Parameter 1 Algoritma Genetika.

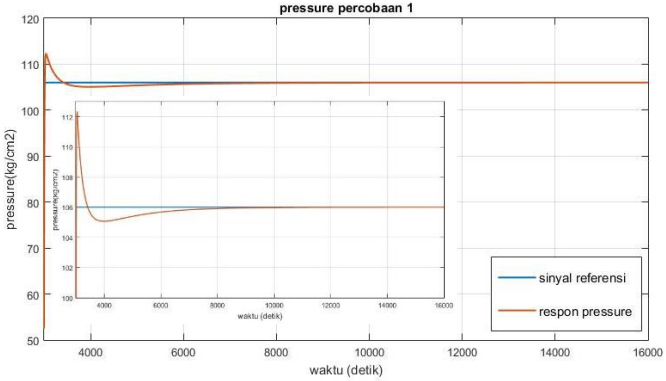
No	Parameter 1	Nilai
1	Populasi (P)	25
2	Generasi (G)	25
3	Rasio Seleksi (Rs)	0,1
4	Rasio *Crossover* (Rc)	0,1
5	Rasio Mutasi (Rm)	0,1

Setelah algoritma genetika dijalankan, perubahan nilai *fitness* individu pada setiap generasi ditunjukkan pada Gambar 4.1.



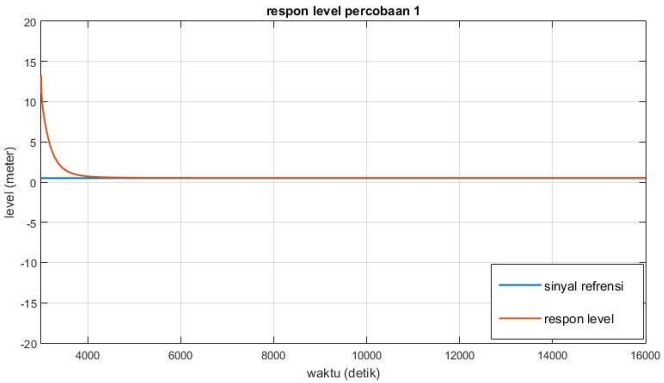
Gambar 4.1 Perubahan *Fitness* Individu Menggunakan Parameter 1.

Pada saat menggunakan parameter 1, algoritma genetika gagal mencapai konvergensi, hal ini dapat dilihat dari perubahan *fitness* terbaik di setiap generasi yang cenderung tetap di sekitar nilai 93,73. Setelah dilakukan *tuning* parameter PID menggunakan parameter 1, didapatkan parameter Kp = 0,1451 Ki = 9,9170e-05 Kd = 4,7844e-14 untuk *pressure* dan Kp = 0,2624 Ki = 7,6734e-09 Kd = 1,3795e-16 untuk *level* dengan nilai *fitness* sebesar 93,73. Parameter hasil *tuning* lalu disimulasikan pada *plant Boiler-turbine* dengan respon *output pressure* dan *level* hasil simulasi digambarkan pada Gambar 4.2 dan Gambar 4.3.



Gambar 4.2 Respon *Pressure Boiler-turbine* dengan Parameter 1.

Saat menggunakan parameter 1, respon *pressure* masih terdapat *overshoot* pada saat transien sebesar 5,8% r*ise time* sebesar 3.010 detik dan *settling time* sebesar 7.000 detik.



Gambar 4.3 Respon *Level Boiler-turbine* dengan Parameter 1.

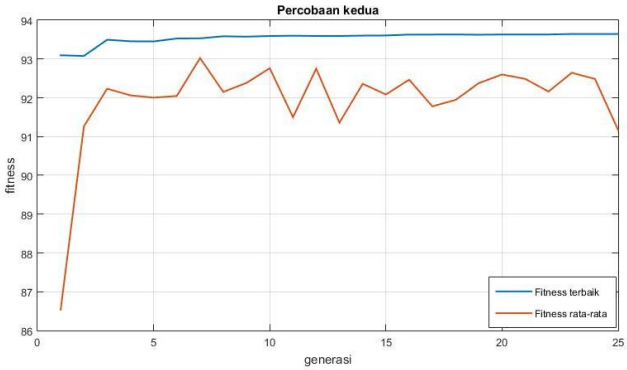
Saat menggunakan parameter 1, respon *level* tanpa *overshoot* dengan *rise time* sebesar 3.400 detik dan *settling time* sebesar 3.600 detik.

Pada percobaan kedua penulis menggunakan parameter 2 yang ditunjukkan pada Tabel 4.2.

Tabel 4.2 Parameter 2 Algoritma Genetika.

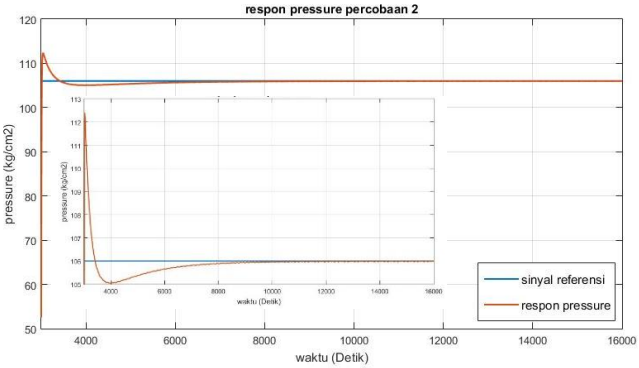
No	Parameter 2	Nilai
1	Populasi (P)	25
2	Generasi (G)	25
3	Rasio Seleksi (Rs)	0,2
4	Rasio *Crossover* (Rc)	0,2
5	Rasio Mutasi (Rm)	0,2

Setelah algoritma genetika dijalankan, perubahan nilai *fitness* individu pada setiap generasi ditunjukkan pada Gambar 4.7.



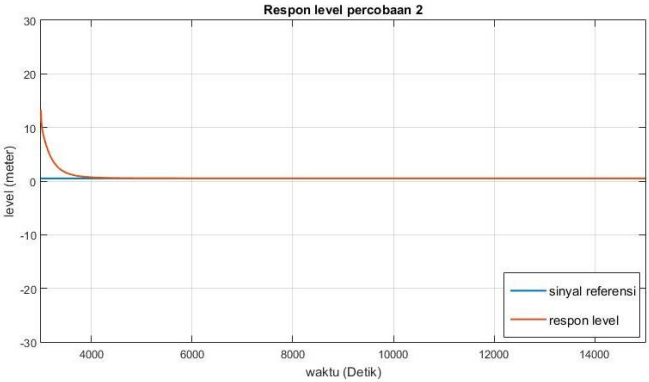
Gambar 4.4 Perubahan *Fitness* Individu Menggunakan Parameter 2.

Pada saat menggunakan parameter 2, algoritma genetika gagal mencapai konvergensi, hal ini dapat dilihat dari perubahan *fitness* terbaik di setiap generasi yang cenderung tetap di sekitar nilai 93,6. Setelah dilakukan *tuning* parameter PID menggunakan parameter 2, didapatkan parameter Kp = 0,1456 Ki = 9,4421e-05 Kd = 3,9282e-13 untuk *pressure* dan Kp = 0,2613 Ki = 9,4924e-09 Kd = 2,2992e-16 untuk *level* dengan nilai *fitness* sebesar 93,6447. Parameter hasil *tuning* lalu disimulasikan pada *plant Boiler-turbine* dengan respon *output pressure* dan *level* hasil simulasi digambarkan pada Gambar 4.5 dan Gambar 4.6.



Gambar 4.5 Respon *Pressure Boiler-turbine* dengan Parameter 2

Saat menggunakan parameter 2, respon *pressure* masih terdapat *overshoot* pada saat transien sebesar 5,9% *rise time* sebesar 3.022 detik dan *settling time* sebesar 7.000 detik.



Gambar 4.6 Respon *Level Boiler-turbine* dengan Parameter 2.

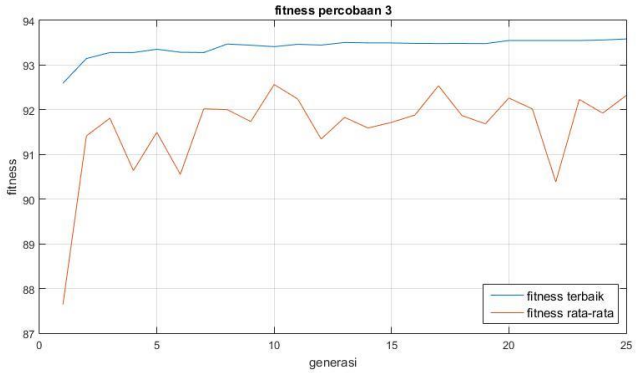
Saat menggunakan parameter 2, respon *level* tanpa *overshoot* dengan *rise time* sebesar 3.400 detik dan *settling time* sebesar 3.600 detik.

Pada percobaan ketiga penulis menggunakan parameter pada Tabel 4.3.

Tabel 4.3 Parameter 3 Algoritma Genetika.

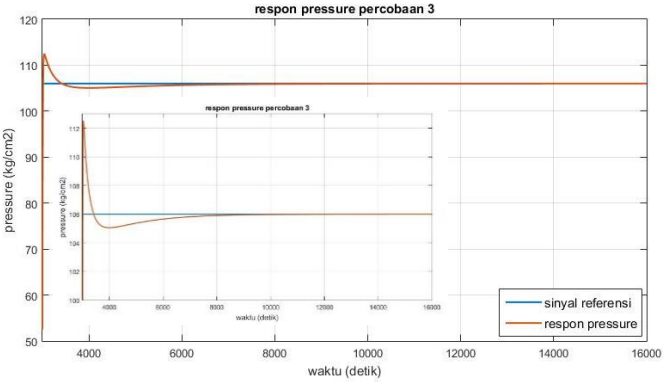
No	Parameter 3	Nilai
1	Populasi (P)	25
2	Generasi (G)	25
3	Rasio Seleksi (Rs)	0,3
4	Rasio *Crossover* (Rc)	0,3
5	Rasio Mutasi (Rm)	0,3

Setelah algoritma genetika dijalankan, perubahan nilai *fitness* individu pada setiap generasi ditunjukkan pada Gambar 4.7.



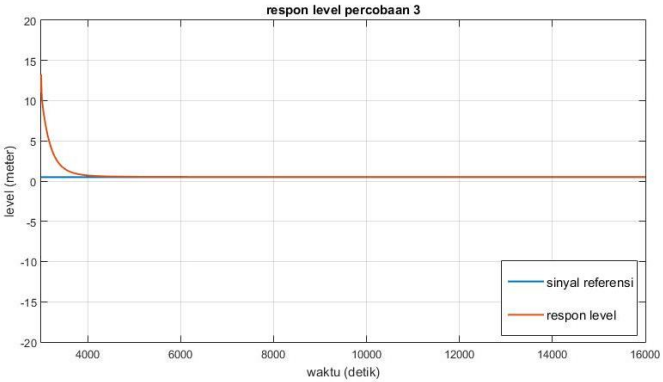
Gambar 4.7 Perubahan *Fitness* Individu Menggunakan Parameter 3.

Pada saat menggunakan parameter 3, algoritma genetika gagal mencapai konvergensi, hal ini dapat dilihat dari perubahan *fitness* terbaik di setiap generasi yang cenderung tetap di sekitar nilai 93,5. Setelah dilakukan *tuning* parameter PID menggunakan parameter 3, didapatkan parameter Kp=0,1426 Ki=9,3656e-05 Kd=5,2599e-14 untuk *pressure* dan Kp= 0,2662 Ki= 8,3667e-09 Kd=4,6014e-16 untuk *level*. dengan nilai *fitness* sebesar 93,58. Parameter hasil *tuning* lalu disimulasikan pada *plant Boiler-turbine* dengan respon *output pressure* dan *level* hasil simulasi digambarkan pada Gambar 4.8 dan Gambar 4.9.



Gambar 4.8 Respon *Pressure Boiler-turbine* dengan Parameter 3.

Saat menggunakan parameter 3, respon *pressure* masih terdapat *overshoot* pada saat transien sebesar 6,4% *rise time* sebesar 3.022 detik dan *settling time* sebesar 9.000 detik.



Gambar 4.9 Respon *Level Boiler-turbine* dengan Parameter 3.

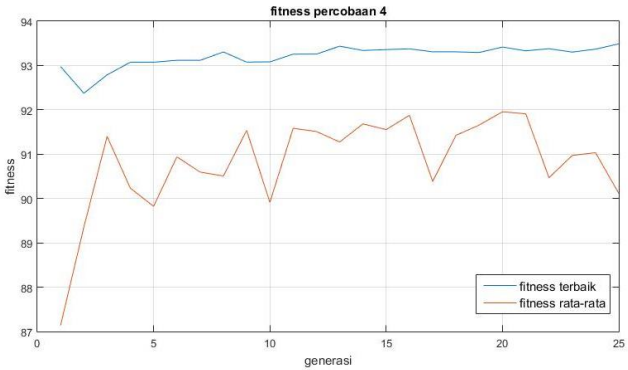
Saat menggunakan parameter 3, respon *level* tanpa *overshoot* dengan *rise time* sebesar 3.600 detik dan *settling time* sebesar 4.000 detik.

Pada percobaan ketiga penulis menggunakan parameter pada Tabel 4.4

Tabel 4.4 Parameter 4 Algoritma Genetika.

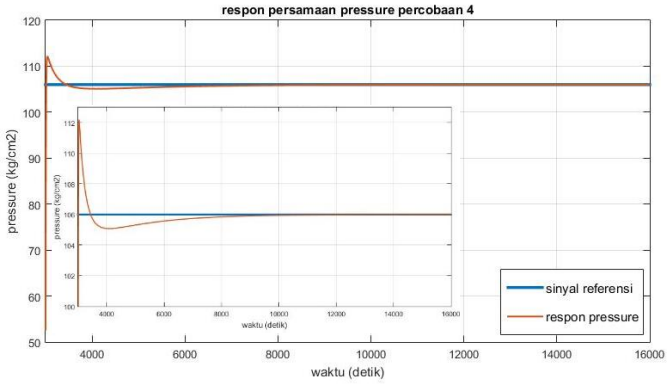
No	Parameter 4	Nilai
1	Populasi (P)	25
2	Generasi (G)	25
3	Rasio Seleksi (Rs)	0,4
4	Rasio *Crossover* (Rc)	0,4
5	Rasio Mutasi (Rm)	0,4

Setelah algoritma genetika dijalankan, perubahan nilai *fitness* individu pada setiap generasi ditunjukkan pada Gambar 4.10.



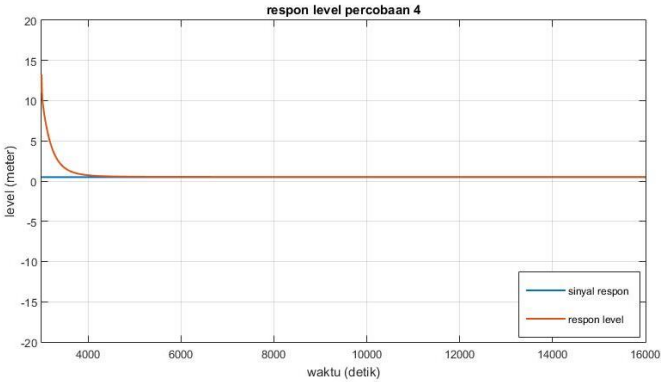
Gambar 4.10 Perubahan *Fitness* Individu Menggunakan Parameter 4.

Pada saat menggunakan parameter 4, algoritma genetika gagal mencapai konvergensi, hal ini dapat dilihat dari perubahan *fitness* terbaik di setiap generasi yang cenderung tetap di sekitar nilai 93,4. Setelah dilakukan *tuning* parameter PID menggunakan parameter 4, didapatkan parameter Kp=0,1447 Ki=7,66776e-05 Kd=3,7261e-13 untuk *pressure* dan Kp= 0,2560 Ki= 1,9491e-09 Kd=8,9912e-16 untuk *level*. dengan nilai *fitness* sebesar 93,48. Parameter hasil *tuning* lalu disimulasikan pada *plant Boiler-turbine* dengan respon *output pressure* dan *level* hasil simulasi digambarkan pada Gambar 4.11 dan Gambar 4.12.



Gambar 4.11 Respon *Pressure Boiler-turbine* dengan Parameter 4.

Saat menggunakan parameter 4, respon *pressure* masih terdapat *overshoot* pada saat transien sebesar 6,8 % *rise time* sebesar 3.022 detik dan *settling time* sebesar 10.000 detik.



Gambar 4.12 Respon *Level Boiler-turbine* dengan Parameter 4.

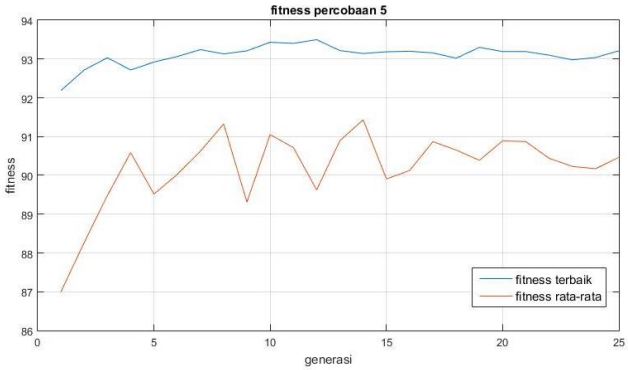
Saat menggunakan parameter 4, respon *level* tanpa *ovetshoot* dengan *Rise time* sebesar 3.600 detik dan *settling time* sebesar 4.200 detik.

Pada percobaan ketiga penulis menggunakan parameter pada Tabel 4.5

Tabel 4.5 Parameter 5 Algoritma Genetika.

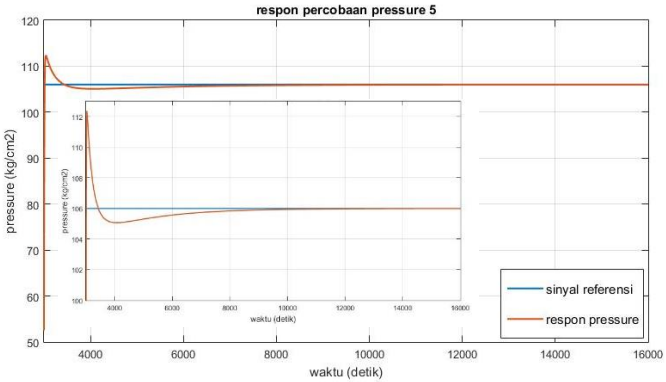
No	Parameter 4	Nilai
1	Populasi (P)	25
2	Generasi (G)	25
3	Rasio Seleksi (Rs)	0,5
4	Rasio *Crossover* (Rc)	0,5
5	Rasio Mutasi (Rm)	0,5

Setelah algoritma genetika dijalankan, perubahan nilai *fitness* individu pada setiap generasi ditunjukkan pada Gambar 4.13.



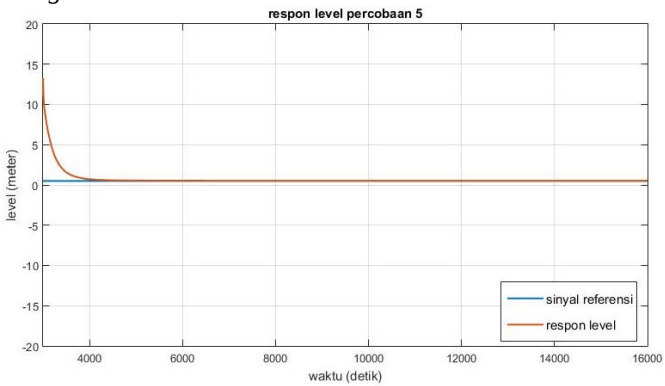
Gambar 4.13 Perubahan *Fitness* Individu Menggunakan Parameter 5.

Pada saat menggunakan parameter 5, algoritma genetika gagal mencapai konvergensi, hal ini dapat dilihat dari perubahan *fitness* terbaik di setiap generasi yang cenderung turun pada nilai 93,2. Setelah dilakukan *tuning* parameter PID menggunakan parameter 5, didapatkan parameter Kp=0,1444 Ki=7,56596e-05 Kd=1,4958e-13 untuk *pressure* dan Kp= 0,2634 Ki= 6,1411e-09 Kd=1,7246e-17 untuk *level*. dengan nilai *fitness* sebesar 93,21. Parameter hasil *tuning* lalu disimulasikan pada *plant Boiler-turbine* dengan respon *output pressure* dan *level* hasil simulasi digambarkan pada Gambar 4.14 dan Gambar 4.15.



Gambar 4.14 Respon *Pressure Boiler-turbine* dengan Parameter 5.

Saat menggunakan parameter 5, respon *pressure* masih terdapat *overshoot* pada saat transien sebesar 6,6 % *rise time* sebesar 3.022 detik dan *settling time* sebesar 11.000 detik.



Gambar 4.15 Respon *Level Boiler-turbine* dengan Parameter 5.

Saat menggunakan parameter 5, respon *level* tanpa o*vershoot* dengan *rise time* sebesar 3.600 detik dan *settling time* sebesar 4.600 detik.

Pada tabel 6 dan tabel 7 dapat dilihat karakteristik *plant pressure* dan *level.* Dimana nilai pada RMSE dan *overshoot* didapat dari hasil tuning GA sedangkan *rise time* dan *settling time* didapat dari respon transien sistem.

Tabel 4.6. Nilai karakteristik *pressure*

Percobaan pressure	RMSE (kg/cm2)	*Overshoot* (%)	*Rise time* (detik)	S*ettling time* (detik)
1	4,9250e+03	5,8%	3.010	7.000
2	4,7199e+03	5,9%	3.022	7.000
3	4,7202e+03	6,4%	3.022	9.000
4	4,7369e+03	6,8%	3.022	10.000
5	4,7354e+03	6,6%	3.022	11.000

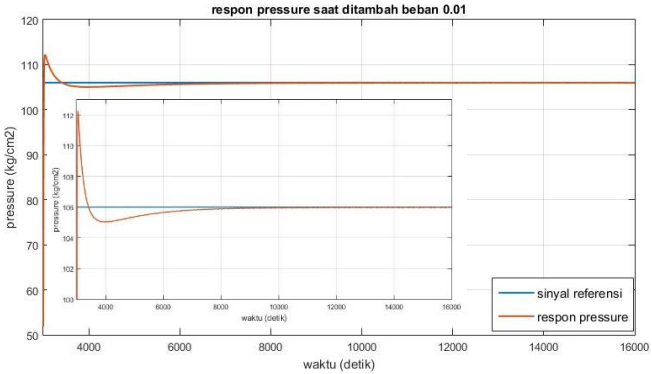
Tabel 4.7. Nilai karakteristik *level*

Percobaan Level	RMSE (meter)	*Overshoot (%)*	*Rise time* (detik)	*Settling time* (detik)
1	311,85	0%	3.400	3.600
2	309,26	0%	3.400	3.600
3	308,23	0%	3.600	4.000
4	310,93	0%	3.600	4.200
5	309,55	0%	3.600	4.600

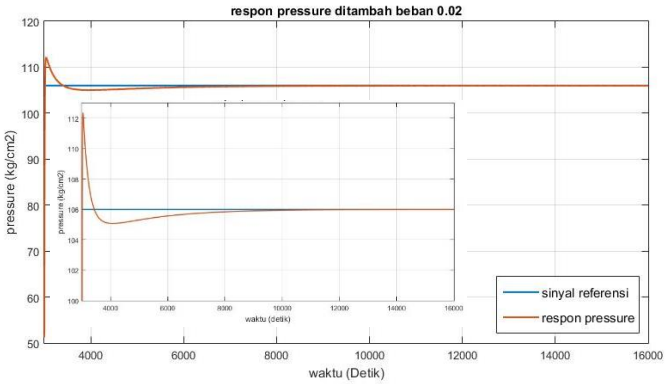
Dari percobaan pertama hingga kelima, nilai *fitness* tidak ada yang dapat mencapai nilai konvergensi, tetapi dengan kelima percobaan tersebut didapat nilai *fitness* terbesar saat percobaan pertama dengan nilai *fitness* 93.78 dengan nilai parameter kontroler PID Kp=0,1451 Ki=9,9170e-05 Kd=4,7844e-14 untuk *pressure* dan Kp=0,2624 Ki=1,3795e-05 Kd=7,6734e-14 untuk *level* .

4.2 Uji Beban

Setelah melakukan pengujian untuk tuning PID dengan algoritma-genetika didapat nilai PID optimal pada percobaan kedua. Lalu, pada tahap ini sistem ditambah beban dengan menambah nilai *steam*. Nilai *steam* awalnya 0,69 dan dinaikkan hingga 0,70 dan 0,71. Dan didapat respon *pressure* setelah diberi beban seperti pada Gambar 4.16 dan Gambar 4.17.

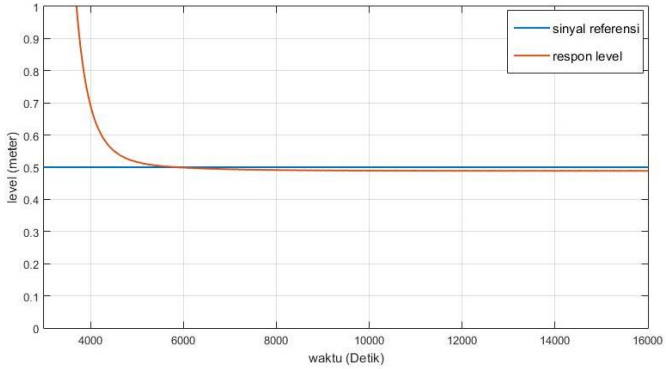


Gambar 4.16 Respon *Pressure* dengan Beban 0,01

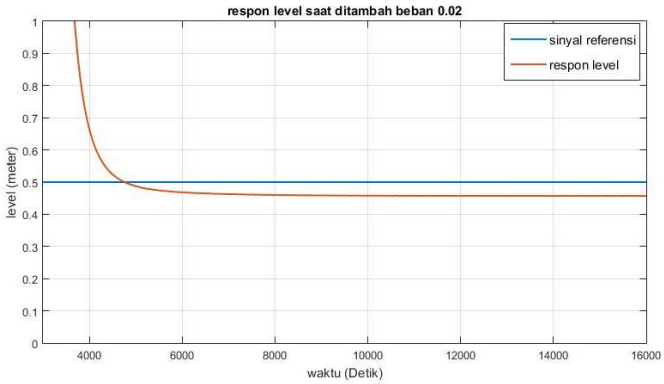


Gambar 4.17 Respon *Pressure* dengan Beban 0,02

Pada respon *pressure* saat ditambah beban 0,01 respon mengikuti sinyal referensi dengan *settling time* 10.000 detik dengan sedikit osilasi. Dan pada respon *pressure* saat ditambah beban 0,02 respon mengikuti sinyal referensi dengan *settling time* 10.000 detik dengan sedikit osilasi. Sedangkan pada respon *level* saat diberi perubahan beban 0,01 dan 0,02 respon seperti pada Gambar 4.18 dan Gambar 4.19



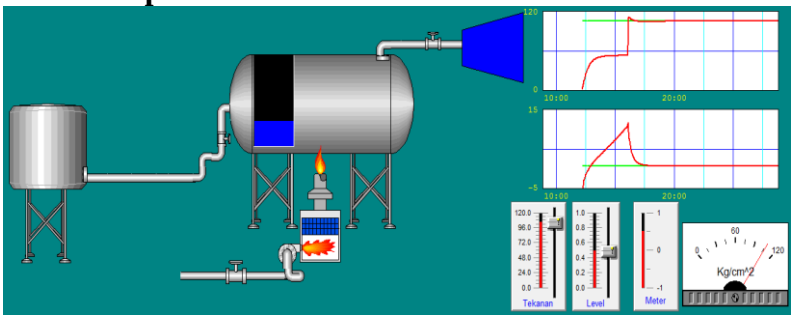
Gambar 4.18 Respon *Level* dengan Beban 0,01



Gambar 4.19 Respon *Level* dengan Beban 0,02

Pada Gambar 4.18 ketika *level* ditambah beban 0,01 maka respon *level* turun dan tidak dapat mengikuti sinyal referensi yang semula 0,5 meter menjadi 0,48 meter dengan *error steady state* sebesar 4%. Sedangkan pada Gambar 4.19 ketika *level* ditambah beban 0,02 respon juga tidak mengikuti sinyal referensi dan cenderung makin turun. Yang semula ketinggian *level* mengikuti *setpoint* pada 0,5 meter menjadi 0,45 meter dengan *error steady state* sebesar 10% .

4.3 Tampilan HMI



Gambar 4.20 Tampilan HMI saat Disimulasikan

Pada tampilan Wonderware yang sudah dikomunikasikan dengan MATLAB dapat dilihat pada Gambar 4.20. ketinggian *level* pada *drum Boiler* ketika dikontrol dapat berada pada kondisi *setpoint* yaitu 0,5 meter.

LAMPIRAN

Program Matlab Algoritma Genetika

Program Utama

```
%Penghitungan dengan menggunakan Algoritma
Genetika!
%================================================
============================!
clear all
clc
Npop=input('Jumlah Populasi=');
jumlah_literasi=input('Jumlah
Generasi/Iterasi=');
rasio_seleksi=input('Rasio Seleksi=');
p_crossover=input('Rasio Crossover=');
p_mutasi=input('Rasio Mutasi=');
literasi=0;
%================================================
============================!
%Pembangkitan populasi awal
for k=1:Npop
    POP(k,1)=(0.1*rand+0.046);      %pembangkitan
nilai random untuk Pp!
    POP(k,2)=(0.0001*rand);  %pembangkitan nilai
random untuk Ip!
    POP(k,3)=(0.000000000001*rand);
%pembangkitan nilai random untuk Dp!
    POP(k,4)=(0.2*rand+0.0663);     %pembangkitan
nilai random untuk Pl!
    POP(k,5)=(0.00000001*rand);     %pembangkitan
nilai random untuk Il!
    POP(k,6)=(0.000000000000001*rand);
%pembangkitan nilai random untuk Dl!
end;
%================================================
============================!
%Mulai iterasi
while literasi<jumlah_literasi
    clc
```

```
    literasi=literasi+1

%================================================
==========================!
    %Penghitungan nilai fitness
    for k=1:Npop;
    Pp=POP(k,1);
    Ip=POP(k,2);
    Dp=POP(k,3);
    Pl=POP(k,4);
    Il=POP(k,5);
    Dl=POP(k,6);

simopt=simset('solver','ode45','srcWorkspace','c
urrent');
    %y=sim('boiler',[],simopt);
    y=sim('nonlinearga',[],simopt);
    in=evalin('base','in');
    out=evalin('base','out');
    %ERMS;
    x=size(in);
    Ex=sum(out-in);
    RMSE=sqrt(Ex^2/x(1));
    %ERMS1;
    x1=size(in1);
    Ex1=sum(out1-in1);
    RMSE1=sqrt(Ex1^2/x1(1));
    %Overshoot
    [t z]=size(out);
    oin=in(t);
    for i=1:t;
        oout=out(i);
        if oout>oin;
            oin=oout;
        end
    end
    os=(oin-in(t))/in(t);
    %Overshoot1
    [t z]=size(out1);
    oin1=in1(t);
```

```
    for i=30000:t;
        oout1=out1(i);
        if oout1<oin1;
            oin1=oout1;
        end
    end
    os1=(oin1-in1(t))/in1(t);
    %nilai fitness

j=0.2*RMSE/10+0.2*RMSE1+0.5*os*100+0.1*os1*100;
    u(k,1)=15000/(1+j);
    end
    POP1=[POP u]
    v=[];
    %  mengurutkan  populasi  berdasarkan  nilai
fitness
    for w=1:Npop
        for x=1:Npop
            if POP1(w,7)>POP1(x,7)
                v=POP1(w,:);
                POP1(w,:)=POP1(x,:);
                POP1(x,:)=v;
            end;
        end;
    end;

%==============================================
============================!
    %Seleksi Truncking dan Mesin Roullet
    [c d]=size(POP1);
    a=floor(c*rasio_seleksi); %pembulatan kebawah
baris*rasio seleksi
    POP2=zeros(a,d);
    for i=1:a
        POP2(i,:)=POP1(i,:);
    end;
    %Seleksi Mesin Roullet
    b=randint(1,c,a)+1;
    for i=1:c
        POP3(i,:)=POP2(b(i),:);
```

```
    end;

%================================================
============================!
    %Crossover
    i=0;
    [a b]=size(POP3);
    anak=zeros(a,b-1);
    while   i<(a-1)   %diiterasi   selama   jumlah
iterasi lebih kecil daripada jumlah populasi)
        i=i+1;
        cut_point=randint(1,1,[1,5]);
        if cut_point==1

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(i+1,1)*
(1-p_crossover)) POP3(i,2:6)];
        elseif cut_point==2

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(i+1,1)*
(1-p_crossover))
(POP3(i,2)*p_crossover)+(POP3(i+1,2)*(1-
p_crossover)) POP3(i,3:6)];
        elseif cut_point==3

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(i+1,1)*
(1-p_crossover))
(POP3(i,2)*p_crossover)+(POP3(i+1,2)*(1-
p_crossover))
(POP3(i,3)*p_crossover)+(POP3(i+1,3)*(1-
p_crossover)) POP3(i,4:6)];
        elseif cut_point==4

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(i+1,1)*
(1-p_crossover))
(POP3(i,2)*p_crossover)+(POP3(i+1,2)*(1-
p_crossover))
(POP3(i,3)*p_crossover)+(POP3(i+1,3)*(1-
p_crossover))
(POP3(i,4)*p_crossover)+(POP3(i+1,4)*(1-
p_crossover)) POP3(i,5:6)];
```

```
        else

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(i+1,1)*
(1-p_crossover))
(POP3(i,2)*p_crossover)+(POP3(i+1,2)*(1-
p_crossover))
(POP3(i,3)*p_crossover)+(POP3(i+1,3)*(1-
p_crossover))
(POP3(i,4)*p_crossover)+(POP3(i+1,4)*(1-
p_crossover))
(POP3(i,5)*p_crossover)+(POP3(i+1,5)*(1-
p_crossover)) POP3(i,6)];
        end;
    end;
    for i=a
        cut_point=randint(1,1,[1,5]);
        if cut_point==1

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(1,1)*(1
-p_crossover)) POP3(i,2:6)];
        elseif cut_point==2

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(i,1)*(1
-p_crossover))
(POP3(i,2)*p_crossover)+(POP3(i,2)*(1-
p_crossover)) POP3(i,3:6)];
        elseif cut_point==3

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(i,1)*(1
-p_crossover))
(POP3(i,2)*p_crossover)+(POP3(i,2)*(1-
p_crossover))
(POP3(i,3)*p_crossover)+(POP3(i,3)*(1-
p_crossover)) POP3(i,4:6)];
        elseif cut_point==4

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(i,1)*(1
-p_crossover))
(POP3(i,2)*p_crossover)+(POP3(i,2)*(1-
p_crossover))
```

```
(POP3(i,3)*p_crossover)+(POP3(i,3)*(1-
p_crossover))
(POP3(i,4)*p_crossover)+(POP3(i,4)*(1-
p_crossover)) POP3(i,5:6)];
        else

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(1,1)*(1
-p_crossover))
(POP3(i,2)*p_crossover)+(POP3(1,2)*(1-
p_crossover))
(POP3(i,3)*p_crossover)+(POP3(i,3)*(1-
p_crossover))
(POP3(i,4)*p_crossover)+(POP3(i,4)*(1-
p_crossover))
(POP3(i,5)*p_crossover)+(POP3(i,5)*(1-
p_crossover)) POP3(i,6)];
        end;
    end;
    POP4=anak;

%================================================
============================!
    %Mutasi
    POP5=POP4;
    jumlah=floor(Npop*6*p_mutasi);       %mencari
jumlah yang akan dimutasi!
    posisi=randint(1,jumlah,[1*6,Npop*6]);
%mencari posisi mana yang akan di mutasi!
    for i=1:jumlah;
        if  mod(posisi(1,i),6)==0;  %nilai  yang
diganti Dl

POP5(posisi(1,i)/6,6)=(0.000000000000001*rand);%
+randint(1,1,[0,10]);
        elseif mod(posisi(1,i),6)==1; %nilai yang
diganti Pp

POP5(ceil(posisi(1,i)/6),1)=(0.1*rand+0.046);%+r
andint(1,1,[0,10]);
```

```
        elseif mod(posisi(1,i),6)==2; %nilai yang
diganti Ip

POP5(ceil(posisi(1,i)/6),2)=(0.0001*rand);%+rand
int(1,1,[0,10]);
        elseif mod(posisi(1,i),6)==3; %nilai yang
diganti Dp

POP5(ceil(posisi(1,i)/6),3)=(0.000000000001*rand
);%+randint(1,1,[0,10]);
        elseif  mod(posisi(1,i),6)==4;%nilai  yang
diganti Pl

POP5(ceil(posisi(1,i)/6),4)=(0.2*rand+0.0663);%+
randint(1,1,[0,10]);
        else %nilai yang diganti Il

POP5(ceil(posisi(1,i)/6),5)=(0.00000001*rand);%+
randint(1,1,[0,10]);
        end
    end;
    POP=POP5;

%==============================================
==========================!
    %Untuk Plot
    iterasi(literasi,1)=literasi;
    topfitness(literasi,1)=POP1(1,7);
    m=sum(POP1,1);
    meanfitness(literasi,1)=m(1,7)/size(POP1,1);
    plot(iterasi,topfitness,iterasi,meanfitness)

%==============================================
==========================!
end;
%menampilkan nilai P, I dan D paling optimal
Pp=POP1(1,1)
Ip=POP1(1,2)
Dp=POP1(1,3)
Pl=POP1(1,4)
```

```
Il=POP1(1,5)
Dl=POP1(1,6)
%================================================
============================!
```

BAB 5
PENUTUP

5.1 Kesimpulan

Dari hasil pengujian sistem kontrol dengan simulasi pada *decoupling* dan kontroler PID tuning algoritma genetika pada plant *boiler* didapatkan beberapa kesimpulan, yaitu:

1. Proses decoupling dapat menghilangkan sifat mempengaruhi antar *input-ouput* pada konfigurasi plant MIMO. Seperti ketika *input* untuk *output pressure* diberi gangguan sinyal random sebesar 10% dari referensinya, maka *output* yang terdapat gangguan hanya pada *output pressure*.
2. Pada lima kali percobaan kontroler PID tuning Algoritma genetika dalam pengendalian *pressure* dan *level plant boiler-turbine*, percobaan pertama dapat memberikan hasil terbaik dengan nilai fitness tertinggi saat diberikan parameter P=25, G=25, Rc=0.1, Rs=0.1, dan Rm= 0.1
3. Dari hasil *tuning* kontroler PID menggunakan algoritma genetika didapatkan parameter optimal sebagai berikut, Kp=0.1451, Ki=9.9170e-05, Kd=4.7844e-14 untuk mengatur pressure dan Kp=0.2624, Ki=7.6734e-09, Kd=1.3795e-17 untuk mengatur *level.*
4. Kontroler PID t*uning* Algoritma genetika mampu mengatasi pembebanan dengan batasan nilai *steam* sebesar 0.71 dengan parameter algoritma genetika.

5.2 Saran

Untuk pengembangan selanjutnya penulis menyarankan agar menerapkan metode dekoling nonlinear atau linearisasi sepotong-sepotong dengan kontroler adaptif di mana metode tersebut diharapkan dapat menanggulangi kenonlinearan *plant* yang mengakibatkan *level* air pada *drum* mempunyai karakteristik yang tidak stabil.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Bell, R.D. and Åström, K.J., "*Dynamic Models for BoilerTurbine-Alternator Units: Data Logs and Parameter Estimation for a 160 MW Unit". Report TFRT-3192*, Lund Institute of Technology, Sweden, 1987.

[2] Tan, W., Horacio, J.M., Chen, T., and Liu, J., "*Analysis and Control of a Nonlinear Boiler-Turbine Unit*", Journal of Process Control 15, pp. 883–891, 2005.

[3] Safrurriza,Mohammad, "Desain Sistem Pengaturan Kecepatan Motor Arus Searah Tanpa Sikat Menggunakan PID Multiobjektif Berdasarkan Algoritma Genetika.", *Tugas Akhir*, Institut Teknologi Sepuluh Nopember, 2016.

[4] Auliya, Mochamad Rasyid, " Perancangan Kontroler PID dan Simulator Human Machine Interface (HMI) pada Sistem Pengaturan Level Minyak Unit Test Separator di Total E&P Indonesia.", *Tugas Akhir*, Institut Teknologi Sepuluh Nopember, 2016.

[5] Al Amin, Anas, "Pengaturn Tekanan Boiler-Turbine Berbasis Hybrid Fuzzy PID.", *Tugas Akhir*, Institut Teknologi Sepuluh Nopember, 2012.

[6] K. J. Astrom and T. Hdgglund, Advanced PID Control, ISA - Instrumentation, Systems, and Automation Society., 2005.

[7] M. Gen and R. Cheng, "*Genetic Algorithm and Engineering Design.*", New York: John Wiley & Sons Inc., 1997.

[8] Choirul, Farid Akbar, "Tuning Parameter Linear Quadratic Tracking Menggunakan Algoritma Genetika utuk Pengendalian Gerak Lateral Quadcopter.", *Tugas Akhir*, Institut Teknologi Sepuluh Nopember, *2016.*

[9] Yuda, Kadek Gus Ermawan, "Perancangan Virtual plant Untuk Pengendalian Drum-Level Boiler Menggunakan Kontroler Linier Quadratic Regulator (LQR).", *Tugas Akhir*, Institut Teknologi Sepuluh Nopember,2010.

RIWAYAT HIDUP

Aryani Fabiany, dilahirkan di Surabaya, Jawa Timur, Indonesia pada tahun 1994. Penulis menempuh pendidikan formal di SD Negeri Ketabang V Surabaya, SMP Negeri 1 Surabaya, SMA Negeri 6 Surabaya. Dan melanjutkan pendidikan tinggi S1 di Teknik Sistem Pengaturan, Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember Surabaya. Email penulis adalah: aryani.fabiany@gmail.com